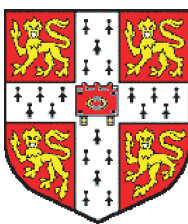


Automated Analysis and Validation of Open Polymer Data

Nicholas William England

St Catharine's College



A dissertation submitted to the University of Cambridge
for the degree of Doctor of Philosophy

Unilever Centre for Molecular Science Informatics
Department of Chemistry
Lensfield Road,
Cambridge, CB2 1EW,
United Kingdom.

February 22, 2011

Disclaimer

This thesis is the result of my own work and includes nothing which is the outcome of work done in collaboration, except where specifically indicated.

This thesis does not exceed the specified word limit (60000) as defined by the Chemistry Degree Committee.

This thesis has been typeset in 12pt font using L^AT_EX2 ϵ according to the specifications defined by the Board of Graduate Studies and the Chemistry Degree Committee.

Abstract

A system to automatically extract, analyse, validate and model polymer data has been produced. This system is called the Polymer Informatics Knowledge System (PIKS).

Methods of storing polymer data electronically are examined. The majority of data-formats are only capable of representing an idealised structure of a macromolecule rather than the actual distribution of structures present in the polymer. Polymer markup language (PML) is the only data-format capable of storing this information. A novel extension to the PML language, allowing copolymers produced with a depletion of reactants is introduced. Without the extension only Markov-chains can be produced.

An informatics analysis of Unilever data of cleaning efficacy of polymers is performed. A representative macromolecule was produced for each polymer sample. Descriptors were calculated over these and used for machine learning to predict the cleaning efficacy. From these models a monomer was identified which was very strongly correlated with good cleaning performance. The monomer in question cannot be revealed as it is a trade secret.

Polymer data from the PoLyInfo database are extracted and converted into XML. A summary of the data available in the PoLyInfo Database is presented. The PIKS tools were used to automatically validate this data for internal consistency, as well as against another data source. The monomers and polymers were analysed for consistency, as well as CML reactions being produced for the polymerisation reactions in the database which were also checked for constancy. The error in the structures was found to be 5.8% for the monomers, 7.3% for the polymers and 2.9% for the reactions. Some of the causes of the discrepancies are presented.

The property data from the PoLyInfo database was then used for machine learning. Support Vector Regression (SVR) models of the glass transition temperature were produced both with and without the inclusion of sample characterisation data. Both methods performed similarly, with the model without producing an RMS error of 19.1K ($r^2 = 0.96$), while the model with produced an RMS error of 20.1K ($r^2 = 0.96$). This means that more sample characterisation data is required than the M_w and M_w/M_n .

Acknowledgments

I would like to thank my supervisor Dr Peter Murray-Rust for guidance and advice during this work. Jerry Winter and Dr Ian Stott for helping me during my time with Unilever. Dr Sam Adams for proof-reading and Java expertise. Dr Nico Adams, Dr Jennifer Ryder, Jim Downing, David Jessop and Dr Lezan Hawizy for useful discussions and advice throughout the PhD. I would also like to thank the UCC computing officers, especially Dr Charlotte Bolton and Dr Philip Marsden. Unilever are thanked for funding.

Special thanks to James Bell, John Mitchell, Max Macaluso, Nick Day, Joe Townsend and Julian Huppert MP for the traditional discussions over tea; CUMPC for keeping me healthy and sane over the years, especially Zoe, KH and Bridget; and Lara for help and support over the years.

Contents

Disclaimer	i
Acknowledgements	i
Table of contents	v
List of tables	vii
List of figures	x
Glossary	xi
1 Introduction	2
1.1 Polymers	3
1.1.1 Representation of Polymers	5
1.1.2 Repeat Unit	6
1.1.3 Variation	6
1.1.4 Characterisation	11
1.1.5 Copolymers	13
1.1.6 Additives	14
1.2 Computer representation of molecules	14
1.2.1 MDL mol	14
1.2.2 SMILES	15
1.2.3 XML	16
1.3 Computer representation of polymers	19
1.3.1 MDL mol for repeat units	19
1.3.2 Polymer Markup Language (PML)	21
1.3.3 PoLyInfo Formula	21
2 Application of Polymer Informatics to Polymer Selection for Concentrated Cleaning	22
2.1 Aim	22
2.2 Background	23

2.3	Predicting Properties	23
2.4	Polymers	24
2.5	Descriptors	26
2.6	Machine Learning Methods	29
2.7	Data	31
2.8	Results and Analysis	32
2.8.1	Soil Class A	32
2.8.2	Soil Class B	36
2.8.3	Soil Class C	36
2.9	Conclusions	37
3	Polymer Markup Language	39
3.1	Introduction	39
3.2	Syntax	40
3.2.1	Core Elements	40
3.2.2	CML	40
3.2.3	ref and id attributes	41
3.2.4	<molecule>	41
3.2.5	Building CML from PML	43
3.3	Example 1	43
3.3.1	<join> Elements	45
3.3.2	countExpression	45
3.3.3	Complete Example	46
3.4	Example 2	48
3.4.1	<fragmentList>	48
3.4.2	<torsion> elements	50
3.5	markushMixture	51
3.6	PolymerBuilder Templates	53
3.6.1	Fixed length homopolymer	53
3.6.2	Statistical copolymer	56
3.7	JUMBO	61
3.8	Extension to PML to allow for depletion of monomer	63
3.8.1	Example usage	65
3.8.2	Comparison with an Experimental Sample	67
3.8.3	Methods	67
3.8.4	Results	69
3.9	Conclusions	69
4	PoLyInfo	72
4.1	History	73
4.2	Content	73
4.3	Navigation	75
4.4	Polymers	81

4.5	Polymer page	82
4.6	The PoLyInfo Formula	84
4.7	Samples of Polymers	88
4.8	Properties	89
4.9	Physical Properties	89
4.10	Optical Properties	92
4.11	Thermal Properties	93
4.12	Electrical Properties	94
4.13	Physicochemical Properties	95
4.14	Dilute solution properties	97
4.15	Monomers	98
4.15.1	JST	98
4.16	Problematic data	100
4.16.1	Kelvin vs Celsius	100
4.16.2	Polydispersity	100
4.17	Aggregate data contained within PoLyInfo	101
4.18	Conclusions	103
5	Polymer Informatics Knowledge System	105
5.1	Objectives	105
5.2	Code for extraction and parsing of PoLyInfo	105
5.2.1	Polyinfo-harvester	106
5.2.2	Property	106
5.2.3	Spider	110
5.2.4	PolyinfoReader	111
5.2.5	Reaction Processor	113
5.3	Xml2Csv	113
5.4	Validation	114
5.5	eXist	115
5.5.1	XQuery	115
5.6	KNIME	116
5.7	OPSIN	117
5.8	Monomer Substructure Search	120
5.9	Conclusions	120
6	Validation of Chemistry in PoLyInfo database	121
6.1	Validation of Monomer data	121
6.1.1	Incorrect Monomer Data	125
6.2	Validation of Polymer Data	131
6.2.1	Incorrect Polymer Data	133
6.3	Validation of the Chemistry	135
6.3.1	Data selection	137
6.3.2	Reactions	138

6.3.3	Results	140
6.4	Conclusions	141
7	Machine Learning Analysis of PoLyInfo Data	143
7.1	Aim	143
7.2	Variation	143
7.3	Cause of Variation	147
7.4	Modelling polymer properties	150
7.4.1	Overview	150
7.4.2	Data	153
7.4.3	Repeat units	154
7.4.4	Descriptors	156
7.4.5	Filtering the Data	157
7.4.6	Support Vector Machine Modelling of Properties	158
7.5	Conclusions	162
8	Conclusions	163
A	XQueries	167
A.1	Monomer ID listings by class	167
A.1.1	XQuery	167
A.1.2	Results	167
A.2	Properties and Experimental Methods	168
A.2.1	XQuery	168
A.2.2	Results	168
A.3	Units	170
A.3.1	XQuery	170
A.3.2	Results	170
B	Descriptors	177
B.1	Physical	177
B.2	Subdivided Surface Areas	178
B.3	Atom Counts and Bond Counts	179
B.4	Kier and Hall Connectivity and Kappa Shape Indices	182
B.5	Adjacency and Distance Matrix Descriptors	183
	Bibliography	185

List of Tables

1.1	Listing for a mol file representing ethanol	15
1.2	CML listings for an ethanol molecule with explicit hydrogens.	19
1.3	mol file listings for poly(ethene) repeat unit.	20
2.1	Table of prediction results for A soil class.	33
2.2	Table of prediction results for B soil class.	36
2.3	Table of prediction results for C soil class.	37
3.1	Catalog.xml listing which allows URIs to be resolved to physical files.	41
3.2	Ethylene oxide CML file.	42
3.3	PML listing for Example 1.	45
3.4	Example 2 PML listing.	49
3.5	Java code to use JUMBO <code>FragmentTool</code> to build a CML document from a PML document.	63
4.1	Table of the different classes of polymers used in the PoLyInfo database.	77
4.2	Table of the <i>popular polymers</i> in the PoLyInfo database.	80
4.3	Table of the different property classes in the PoLyInfo database.	91
4.4	Table of the different monomer classes along with their number of entires as defined in the PoLyInfo database.	99
5.1	An example truncated <code><Polymer></code> element	109
5.2	Example XML for a monomer in PIKS eXist database.	110
5.3	Example XML for JST data in PIKS eXist database.	111
5.4	BNF-like Antlr grammar for parsing PoLyInfo formula	112
5.5	Example XQuery which lists monomers by class.	116
6.1	Table of the duplicated JST numbers in the PoLyInfo database.	123
6.2	Table of the different classes defined in the PoLyInfo database.	136
6.3	Table of the reactions in PoLyInfo database	141

7.1	The most common polymers in the PoLyInfo database by sample count.	144
-----	--	-----

List of Figures

1	Overview of the Polymer Informatics Knowledge System (PIKS) produced for this work. This system was built for the project and is described in Chapter 5. PIKS is used in Chapters 3, 4,6 and 7.	1
1.1	Step-growth and chain growth polymerisation	4
1.2	Kinetics of step-growth vs living chain-growth	5
1.3	Ethene and poly(ethene)	6
1.4	Repeat unit representation	7
1.5	Both ethene and diazomethane are monomers which can be used to form the same repeat unit.	7
1.6	Mechanism for branching a polymer chain via backbiting. . . .	8
1.7	Termination via coupling and disproportionation.	10
1.8	The meso and racemo dyads for poly(prop-1-ene).	11
1.9	A diagram to show the repeat unit of poly[ethene-co-(prop-1-ene)].	14
2.1	Descriptors scaling with repeat units	27
2.2	The process of SciTegic fingerprint generation. Initially each atom is assigned a number based on its element and number of bonds. This contains the information shown for iteration 0. During each iteration, the numbers of each neighbouring atoms are appended to the number of the original atom and hashed to produce a new number. This adds the information from the previous iteration of each of an atom's neighbours. In iteration 2, the information from an atom's neighbours in iteration 1 is added. The process for the carbon atom marked with the arrow is shown.	28
2.3	ROC plot for A soil class CART model.	33
2.4	Depiction of CART tree for A soil class	34
2.5	Demonstration substructure	35
3.1	Representation of the eo molecule	43

3.2	A diagram showing a representation of the three fragments used in Example 1.	44
3.3	The expansion of a countExpression attribute	46
3.4	2D representation of the process of joining the molecule in Example 1 and the completed molecule.	47
3.5	Representation of the eocl fragment from line 10 in example 2 (Table 3.4)	50
3.6	Representation of complete molecule from Example 2 (Table 3.4)	50
3.7	How a torsion angle can be defined between two fragments that are joined together	52
3.8	Screenshot of the HTML interface for the polymer builder web-service. This is the template for a fixed length homopolymer. .	54
3.9	The process of building a CML document from the HTML form.	55
3.10	A 3D representation of an example molecule created using the fixed length homopolymer template.	56
3.11	Screenshot of the HTML interface for the statistical copolymer template.	57
3.12	Some possibilities for how the AX <fragment> can be expanded recursively.	62
3.13	3D representation of a statistical co-polymer	62
3.14	Transformation of data to be comparable to GPC	68
3.15	Contrast of experimental characterisation data with PML . . .	70
4.1	Data in PoLyInfo over time	74
4.2	Diagram from the PoLyInfo website describing the data-structure used in the database.	76
4.3	The navigation sidebar menu from the PoLyInfo database. . .	78
4.4	The basic search menu from the PoLyInfo database.	81
4.5	A typical polymer page from the PoLyInfo database.	83
4.6	Example PoLyInfo formula	86
4.7	The structure of kevlar.	87
4.8	The structure of poly(oxy-1,4-phenylenesulfonylnaphthalene-2,7-diylsulfonyl-1,4-phenylene)	88
4.9	An example sample page. In this case the characterisation data for the sample contain the processing information, the crystallinity, stereoregularity and the name of the manufacturer.	90
4.10	Distribution of properties per sample	91
4.11	The distribution of temperatures at which density measurements were made.	92
4.12	Conditions of dielectric constant measurements.	96
4.13	Distribution of temperatures for conductivity measurements .	97
4.14	Scatter plot of T_g vs T_m in PoLyInfo database	102

5.1	The different tools and code used to download and extract information from the PoLyInfo Database	107
5.2	The KNIME workflow for loading in the CSV file produced by Xml2Csv	118
5.3	This KNIME workflow takes the cleaned up CSV file, and splits it into a training, validation and test set	119
6.1	The process of validation of the monomer data.	124
6.2	Summary of monomer validation failure	125
6.3	OPSIN interprets ketones different to ChemDraw and PoLyInfo	126
6.4	Incorrect name for monomer	128
6.5	ChemDraw produces wrong isomers	130
6.6	ChemDraw produces wrong structure: ketone	131
6.7	ChemDraw produces wrong structure: phosphinate	131
6.8	The process of validation of the polymer data.	132
6.9	Incorrect name for polymer	134
6.10	Ambiguous name for polymer	134
6.11	Examples where a repeat unit is half the size of the monomer	137
6.12	The process used to select good reactions	138
6.13	Nylon 6 polymerisation	139
6.14	Diagram to show the classification of the reaction data.	139
6.15	Poly(methyleneoxypentamethylene oxide), an example of a possible double which is not a double	140
6.16	An example bad reaction	140
7.1	Boxplot of glass transition temperature for common polymers	145
7.2	The variation in glass transition temperature between all the points in the database.	146
7.3	The variation in glass transition temperature between all the points in the database shown as a histogram.	147
7.4	Density of Ethene over time	148
7.5	Density of poly(methyl methacrylate) over time	149
7.6	Workflow for the prediction	151
7.7	Powerlaw of samples of polymers	152
7.8	Histogram of samples per polymer	152
7.9	Pentamer of poly(styrene)	155
7.10	Glass transition temperature models correlation	160
7.11	Glass transition temperature models error	161

Glossary

AUC Area Under the Curve

BNF Backus-Naur Form

CART Classification And Regression Tree

CAS Chemical Abstracts Service

CIC Complementary Information Content

CML Chemical Markup Language

CSV Comma Separated Values

DMA Dynamic Mechanical Analysis

DSC Differential Scanning Calorimetry

GPC Gel Permeation Chromatography

HT Head-to-Tail

HTML HyperText Markup Language

IUPAC International Union of Pure and Applied Chemistry

JST Japan Science and Technology Agency

KNIME Konstanz Information Miner

MALDI-TOF Matrix-Assisted Laser Desorption Ionization-Time of Flight

MOE Molecular Operating Environment

NIMS National Institute of Material Sciences

NMR Nuclear Magnetic Resonance

PIF PoLyInfo Formula

PIKS Polymer Informatics Knowledge System

PML Polymer Markup Language

RBF Radial Basis Function

RMS Root Mean Square

ROC Receiver Operating Characteristic

SEC Size-Exclusion Chromatography

SMILES Simplified Molecular Input Line Entry Specification

SQL Structured Query Language

SVM Support Vector Machine

SVR Support Vector Regression

WEKA Waikato Environment for Knowledge Analysis

XHTML eXtensible HyperText Markup Language

XML eXtensible Markup Language

XSLT Extensible Stylesheet Language Transformations

Polymer Informatics Knowledge System

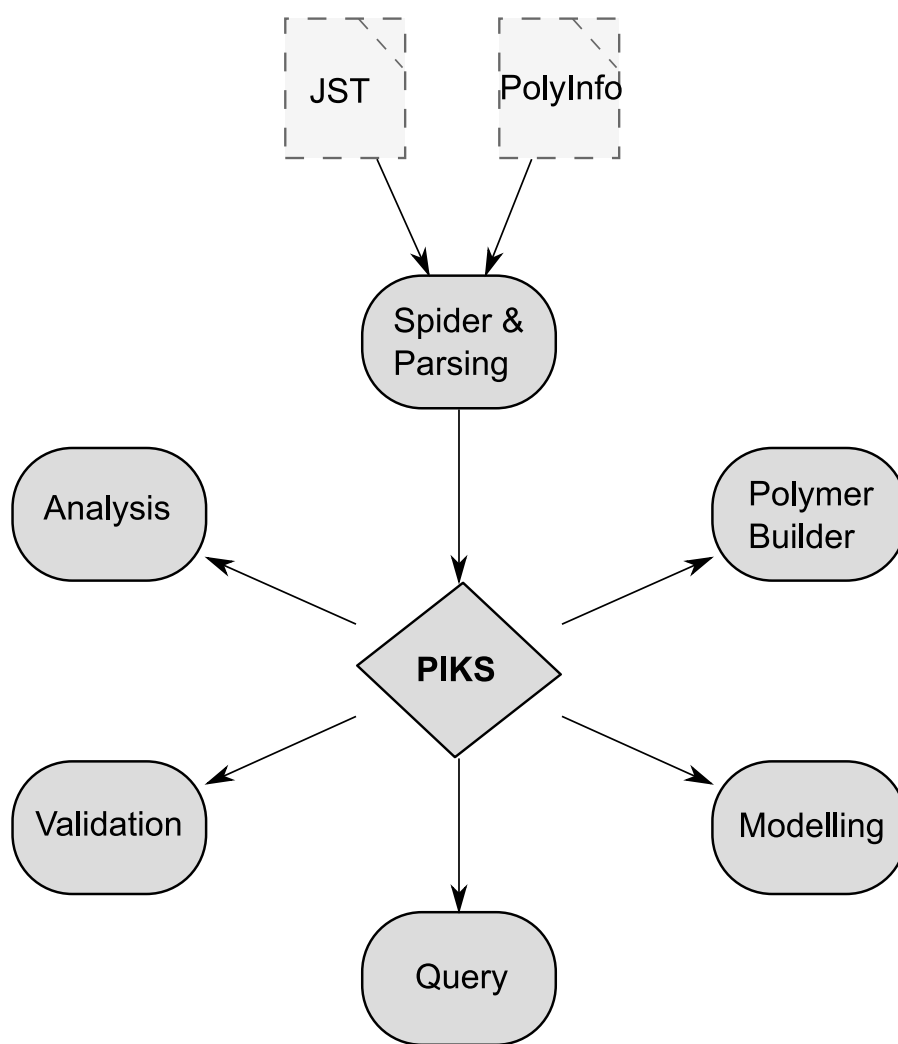


Figure 1: Overview of the Polymer Informatics Knowledge System (PIKS) produced for this work. This system was built for the project and is described in Chapter 5. PIKS is used in Chapters 3, 4, 6 and 7.

Chapter 1

Introduction

The quantity of scientific data being published far exceeds the human capacity to process it. In 2009 alone, 42,847 papers were published on polymers with 18,567 in the domain of chemistry¹. Computerised methods for interrogating and using the existing data is therefore necessary. Informatics techniques have become increasingly commonplace in the fields of drug-design[1, 2] and chemistry[3], but little progress has been made on polymers[4].

Polymers are used every day by the vast majority of people. Everything from shampoo and detergent to plastic bags and clothes gets made from synthetic polymers. Experiments to create new polymers and determine their properties are expensive. If the properties of novel polymers can be predicted by computational methods, then resources can be focused on synthesising polymers with the desired properties. Prediction of polymer properties by computational methods is therefore a desirable objective. Before this can be achieved, it is necessary to have a knowledge resource of the existing polymer literature data with which to train the models.

In Chapter 2 a project undertaken with Unilever on the analysis of cleaning efficacy data for polymers is described. In this chapter the use of machine learning techniques on polymer data to predict cleaning effectiveness is discussed. Due to trade secrets some structural details of the polymers are omitted.

¹Numbers were obtained using the Web of Knowledge

PML (Polymer Markup Language) is a format for storing structural polymer data. This format is described in Chapter 3 as well as a novel extension to the language which allows additional types of polymer to be represented. The structural data held within the PoLyInfo database is in a propriety format. A tool to convert these structures into PML is described in Chapter 5.

A publicly available repository of polymer data is the PoLyInfo database. This is described in Chapter 4. Tools have been developed, which together make up the Polymer Informatics Knowledge System (PIKS) shown in Figure 1, to facilitate the automatic extraction of polymer data from the database for analysis, validation, querying and use in machine learning. These tools are described in Chapter 5.

In Chapter 6 the automatic validation of monomer, polymer and reaction data is discussed. This is performed using the validation component of PIKS. The tools that have been developed in order to facilitate this validation of the monomer, polymer and reaction data are described here. Finally in Chapter 7 the modelling component of PIKS is used for machine learning analysis of a polymer property.

In this Chapter the essentials of polymer structure is discussed. This includes how polymer structures are represented on paper and electronically, and details of how different polymer samples of the same polymer can vary in structure, which gives rise to property variation.

1.1 Polymers

Polymers are substances which consist of aggregations of macromolecules. A macromolecule is defined by the International Union of Pure and Applied Chemistry (IUPAC) Gold Book[5] as:

“A molecule of high relative molecular mass, the structure of which essentially comprises the multiple repetition of units derived, actually or conceptually, from molecules of low relative molecular mass.”

A polymer does not normally consist of identical macromolecules. In any particular polymer sample there will be a vast number of different macro-

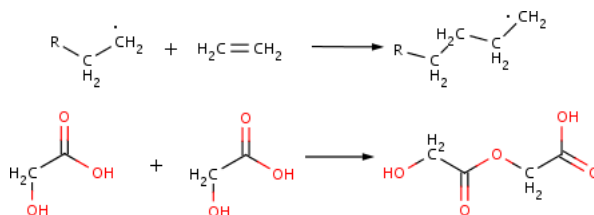


Figure 1.1: Step-growth and chain growth polymerisation reaction. The top reaction is chain-growth reaction, while the bottom reaction is step-growth. A living chain-growth polymerisation is one where no termination reactions occur.

molecules. (Except for some naturally occurring polymers such as proteins, which are outside the scope of this work). The cause of these differences are discussed in Section 1.1.3. This contrasts with small molecule chemistry where a sample of a chemical contains only a single species. This disconnect between the name of a sample and its contents causes additional challenges when dealing with polymers compared to small molecules.

Polymers are manufactured in a polymerisation reaction from one or more monomers. There are a wide variety of different types of reaction for producing polymers. These can be divided into two classes, *chain-growth polymerisation* and *step-growth polymerisation*[6]. In chain-growth polymerisation an initiator causes a number of chains to start growing. The initiators are either free-radical, anionic or cationic. These chains then react with monomers to produce longer chains. Eventually the chains will either terminate or run out of monomers. An example of a polymer produced through this type of reaction is poly(ethene).

In step-growth polymerisation the monomers can all react as they have reactive functional groups. The monomers react to form dimers, which in turn react to form tetramers. The reaction continues until one of the reactants is exhausted, or the solution solidifies. An example of a polymer produced through this type of reaction is poly(glycolic acid) (Figure 1.1). These two different mechanisms have key differences kinetically, since in a chain-growth reaction the length of a chain grows linearly with number of reaction steps, while for step-growth the length of a macromolecule grows exponentially with the number of reaction steps. This difference is shown in

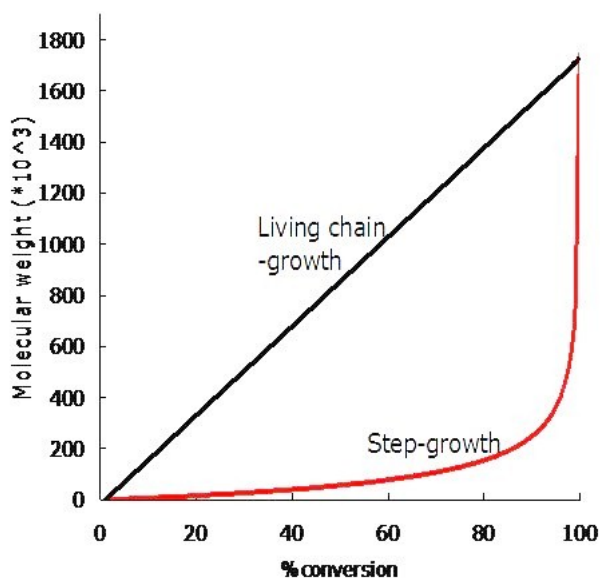


Figure 1.2: Difference in kinetics between a step-growth and a living chain-growth polymerisation. A living chain-growth polymerisation is one where there are no termination events.³

Figure 1.2.

1.1.1 Representation of Polymers

A range of methods for representing polymers have been developed. These include a name, a visual depiction of the repeat unit, a computer readable repeat unit and other computer readable formats. For some polymers, where the structure is unknown, all that can be described is the monomer or monomers that were used to produce it.

In Figure 1.3 two molecules are shown. On the left is an ethene molecule. The word ethene is always represented by this structure. On the right is an example poly(ethene) molecule. A sample of poly(ethene) might contain this molecule, but will also contain a large number of other molecules with varying branching and chain length. The word poly(ethene) therefore represents an ensemble of macromolecules. There are two types of polymer name, *source-based*, which derive solely from the monomers used and *structure-based*, which derive from the repeat unit of the polymer. The IUPAC structure-based

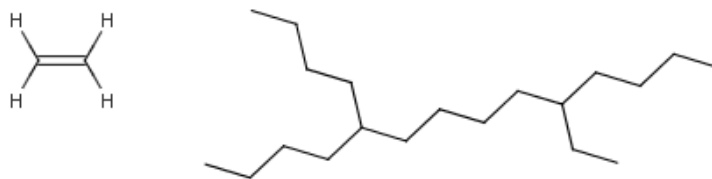


Figure 1.3: The molecule on the left is ethene. The word ethene refers exclusively to this molecule. The molecule on the right is a small poly(ethene) molecule. Poly(ethene) does not exclusively refer to this particular molecule as the degree of branching and length of chain can vary dramatically.

name for poly(ethene) is poly(methylene), which is based on the repeat unit. Polymer repeat units are described in Section 1.1.2.

1.1.2 Repeat Unit

In order to represent the ensemble of macromolecules, a polymer is often represented by a repeat unit (Figure 1.4). A repeat unit has two or more bonds extending outside the square brackets. This indicates that these bonds should connect to the previous and next identical repeat units to create an infinite chain. The actual macromolecules in a polymer sample will not be infinite, and will have end groups which will be different to the repeat unit. Two repeat units for poly(ethene) are shown in Figure 1.4. The repeat unit on the left more closely matches the common monomer, ethene, while the repeat unit on the right more closely matches another monomer, diazomethane. These monomers are shown in Figure 1.5. It is common practise to use the repeat unit with the smallest number of atoms to depict the polymer. IUPAC guidelines specify that the smallest repeat unit should be used. [7]

1.1.3 Variation

As previous mentioned, a polymer sample contains an ensemble of macromolecules. The structure of these macromolecules can vary wildly both between polymer samples and within the ensemble of macromolecules that form

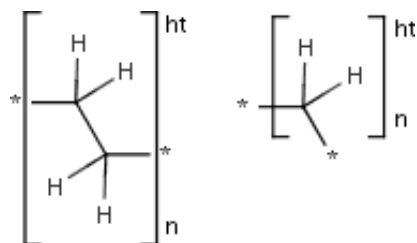


Figure 1.4: These two repeat units both represent poly(ethene). The one on the left shows two CH₂ groups while the one on the right shows only one. Both structures represent the same macromolecule as *n* is considered to be infinite. The repeat unit on the left cannot represent an odd number of CH₂ groups, while the one on the right can. It is common practise to represent a polymer with the shortest repeat unit possible. In this case this is the repeat unit on the right. The ht superscript on the square brackets indicates that the repeat units are joined together head-to-tail.



Figure 1.5: Both ethene and diazomethane are monomers which can be used to form the same repeat unit.

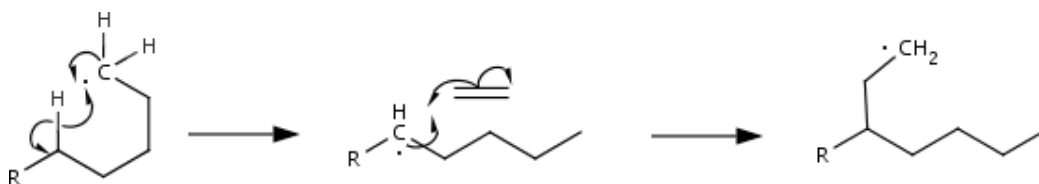


Figure 1.6: Branching can occur when the radical at the end of a growing chain abstracts a hydrogen from its own chain. This is known as *backbiting*. The resulting chain still has a radical and so will undergo further growth, but the chain now has a short side-chain known as a branch. Branching can also occur if a hydrogen is abstracted from a different polymer chain. This will result in a much larger side-chain.

a single polymer sample. The different types of structural variation; namely branching, chain length, tacticity and terminal groups; the how these variations arise and the effects on the properties of the polymer are discussed below:

Branching

While the repeat unit may indicate a linear chain, some polymers can have branching reaction steps. Branches often arise during chain-growth polymerisation, but also arise from step-growth polymerisation when the monomers contain more than two reactive groups.

In free-radical polymerisation, branching occurs when the free-radical at the end of a chain abstracts a hydrogen. If the hydrogen is from its own chain the result is a short side-chain, known as *backbiting*. (Figure 1.6). Hydrogen abstraction can also occur between chains, which results in a much longer side-chain as one macromolecule is now attached to the other.

Branching affects the physical properties of the polymer. In a sample with a large degree of branching, the macromolecules are prevented from packing as closely as the non-branching polymer. The greater the degree of branching, the lower the density of resulting polymer.

Molecular weight

The length of a macromolecule determines its molecular weight. As a polymer is an ensemble of macromolecules the bulk properties of the polymer depend on the distribution of molecular weights of macromolecules that make up the polymer sample.

The molecular weight distribution of a polymer will depend on the method of polymerisation used, impurities in the monomers and reaction conditions. In order to summarise the molecular weight distribution of a polymer, a range of average molecular masses are used. The most common are *number average molar mass* (M_n), *weight-average molecular weight* (M_w) and *viscosity average molar mass* (M_v).

These averages are defined below:

$$M_n = \frac{\sum M_i N_i}{\sum N_i}$$

$$M_w = \frac{\sum M_i^2 N_i}{\sum M_i N_i}$$

$$M_v = \left[\frac{\sum M_i^{1+a} N_i}{\sum M_i N_i} \right]^{1/a}$$

where N_i is the number of molecules having molecular mass M_i and a is the constant from the *Mark-Houwink-Sakurada equation* which varies with polymer, solvent and temperature. The Mark-Houwink-Sakurada equation links viscosity to average molecular mass:

$$[\eta] = K M_v^a$$

where $[\eta]$ is the intrinsic viscosity and K is a constant.

When a measurement of the average molecular mass is conducted, depending on the method used, one of these averages is given as the result. For example, if the intrinsic viscosity is measured, M_v can be calculated. If light-scattering experiments are conducted then M_w is the result. M_n can be obtained from freezing point suppression or end-group analysis methods.[8]

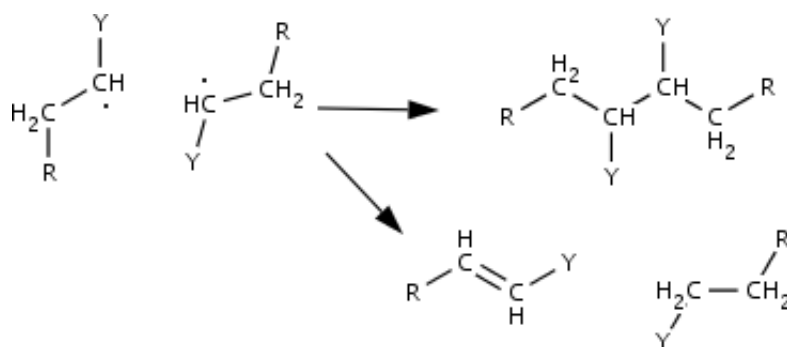


Figure 1.7: Termination can occur with coupling (shown on the top right) or with disproportionation (shown on the bottom right.)

In order to characterise the variation of the molecular mass of the sample, a metric called *polydispersity* is used. This is defined as the ratio of M_w and M_n . A polydispersity of 1 indicates that the sample consists solely of macromolecules with identical molecular weight. Common values for M_w/M_n are between 1 and 3.

Termination reactions

A chain-growth polymerisation generally ends in termination. Termination can either occur with *coupling*, where two chain ends join head-to-head, or *disproportionation*, where an atom such as hydrogen is transferred between two chain ends. (Figure 1.7). Both processes terminate two chains. In the case of coupling one chain is produced, while when disproportionation occurs two chains are produced. This will affect the molecular weight distribution, since chains produced by coupling have a molecular weight equal to the sum of the two chains. In the case of coupling the two chains are joined head-to-head with an initiator at each end. For disproportionation there will be an initiator at just one end.

Tacticity

When a chiral monomer is added to a polymer chain it can be added with the same, or opposite, stereo configuration as the previous monomer. If all the monomers are added in the same configuration the resulting macromolecule

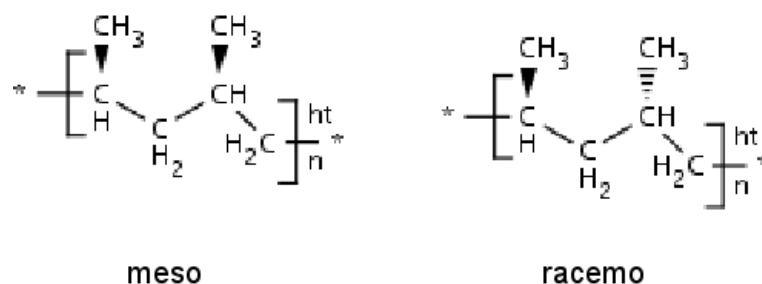


Figure 1.8: The meso and racemo dyads for poly(prop-1-ene).

is described as *isotactic*. If the monomers are all added with the opposite configuration the macromolecule is described as *syndiotactic*. If there is a random distribution then the macromolecule can be described as *atactic*. The tacticity of a polymer sample will depend on the polymerisation method used. For example, poly(prop-1-ene) has a chiral centre in the repeat unit. Depending on the method of polymerisation samples of poly(prop-1-ene) can be either isotactic, syndiotactic or atactic.

Tacticity can be quantified by classifying each pair of repeat units as either *meso* or *racemo* dyads if they have the same or opposite chirality respectively (Figure 1.8). Triads, Tetrads and higher order polyads are also used to describe the tacticity. This allows for a quantitative measure of tacticity.

Tacticity is important as polymer properties such as crystallinity and glass transition temperature very depending on the tacticity of the polymer. In general atactic polymers are less crystalline than the more ordered syndiotactic or isotactic polymers.

1.1.4 Characterisation

A range of techniques exist to characterise the variations in a polymer sample. These include Size-Exclusion Chromatography (SEC), matrix-assisted laser desorption ionization-time of flight mass spectrometry (MALDI-TOF) and nuclear magnetic resonance (NMR).

Size-exclusion chromatography

Size-exclusion chromatography (SEC) is used to determine the molecular weight distribution of a polymer sample. SEC is also known as Gel Permeation Chromatography (GPC). A polymer sample is dissolved in solvent and passed through a column. The column is filled with porous beads that possess a variety of pore sizes. This means that the smaller macromolecules can diffuse into a large number of pores, while the largest macromolecules can only diffuse into a small number of pores. This means that the largest macromolecules elute first, with the smallest eluting last. This technique measures hydrodynamic volume rather than directly measuring molecular weight. The elution volume is proportional to the log of the molecular weight. A range of calibration techniques exist to convert the elution volume into molecular weight, either using a sample with a well-characterised molecular weight distribution or using MALDI-TOF.

MALDI-TOF


In MALDI-TOF mass spectroscopy the polymer sample is suspended in a light-absorbing matrix. A laser pulse is then used to transfer energy to the matrix which in turn transfers it to the macromolecules in the polymer sample. This process vaporises the macromolecule and ionises it. A powerful electric charge is used to accelerate the ion to hit a detector. The time of flight is used to calculate the mass of the ion, since the force is the same for all singly-charged ions.


NMR


Tacticity can be measured using NMR spectroscopy. The relative ratios of the meso and racemo dyads or higher order polyads can be measured from the integrals of the peaks in the NMR spectrum, allowing the tacticity of the sample to be elucidated.


1.1.5 Copolymers

It is common for a polymer to be synthesised from two or more monomers. The IUPAC definition of a copolymer is “*A polymer derived from more than one species of monomer.*” Copolymers can be divided into two classes: those which can be represented by a single repeat unit, and those which cannot. In the first case the copolymer has a fixed overall repeat unit. In the second case the overall repeat unit will contain a repeat unit from each monomer; the polymer has no fixed structure, with some distribution of the individual repeat units distributed along the chain. Common distributions are given below:

Alternating Copolymer In this case the two monomers alternate. This gives rise to one overall repeat unit. A large number of step-growth polymers are alternating copolymers. 

Block Copolymer In this case the macromolecules consist of a large block of one repeat unit, followed by a large block of another repeat unit. There can be more than two blocks, and a block can itself consist of a copolymer. 

Random Copolymer Here the repeat units are connected randomly with no defined order. 

Statistical Copolymer In a statistical copolymer there is some distribution ordering the structure. In a chain-growth polymer the probabilities of different monomers being added depends on the previous monomer added to the chain. 

An example repeat unit for a copolymer, poly[ethene-co-(prop-1-ene)] is shown in Figure 1.9. The *ran* subscript is used to indicate that it represents a random copolymer. The subscripts *alt*, *block*, and *stat* can be used to represent alternating, block, random and statistical copolymers respectively.

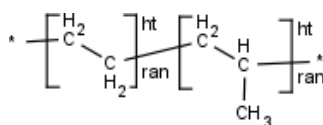


Figure 1.9: A diagram to show the repeat unit of poly[ethene-co-(prop-1-ene)].

1.1.6 Additives

Polymer samples often contain additives. These are small molecules which are added to the polymer sample to change the physical properties of the polymer. They affect the properties in both the liquid and solid state. They may be added to change the polymer properties during processing, or to influence the properties of the finished product. In some cases additives are used to simply “bulk out” the polymer for economic reasons.[8]

Additives are added for a huge variety of reasons and to obtain different effects. The most common additives are plasticisers which are used to reduce the melt viscosity to increase ease of processing. For the purposes of this study polymer samples without additives have been studied.

1.2 Computer representation of molecules

In order to use computational methods, it is necessary to represent the structure of molecules in a form computers can understand. A range of file formats have been developed and used over the years. A number of common formats for representing molecular structure are described, along with their extensions for representing polymer structures.

1.2.1 MDL mol

The mol file format is used by a large number of computer programs and systems. The mol format is one of several very similar formats developed by MDL from 1979 [9]. While originally developed for use by MDL products, they became widespread and are used by a large number of different software programs. The format consists of fixed width columns of data which

```

1 Ethanol
2 Marvin 08161014082D
3
4 3 2 0 0 0 0 999 V2000
5 1.8562 3.3295 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0
6 2.5707 3.7420 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0
7 3.1541 3.1586 0.0000 O 0 0 0 0 0 0 0 0 0 0 0 0
8 1 2 1 0 0 0 0
9 2 3 1 0 0 0 0
10 M END

```

Table 1.1: Listing for a mol file representing ethanol. Line numbers have been added for ease of reference.

are nearly indecipherable without reading the specification. The format has limited capacity for extensions to contain additional information.

A mol file for ethanol is shown in Table 1.1. The first line is the name of the molecule. The second line contains metadata for the time of creation and the program used to create it. On line 4 the number of atoms and bonds in the molecule are declared, with the subsequent 3 lines declaring the x,y co-ordinates of atoms and their element and the next 2 lines listing the bonds between atoms. The twelve columns containing “0” are used to contain information about isotopes, charges and other features of the atom. Without recourse to the format specification it is challenging for a human to read. While it may be possible to add an extra column for in-house tools, if two different groups added an extra column it would be impossible to reconcile the extensions to the format. The vast majority of existing tools will throw an error if the number of columns does not match the specification.

The mol format includes some additions for representing polymer repeat units. These are discussed in Section 1.3.1. The maximum size of molecule which can be represented in the normal mol format is 999 atoms. Using the addition for representing repeat units, a repeat unit of up to 999 atoms can be represented.

1.2.2 SMILES

Simplified molecular input line entry specification (SMILES)[10] is a chemical structure file format designed to be human-readable. SMILES is a line-

format, which means a chemical structure is stored in one string with no line breaks. The basic syntax of SMILES is straightforward; letters represent atoms that are bonded to the preceding and following atom. Branches are achieved using parenthesis and number pairs are used to mark ring closures. Hydrogens are generally omitted and are present implicitly. Standard valences are used to fill in missing hydrogens from the SMILES string. For example carbon has a standard valency of four, and so if a carbon atom only has two bonds, it is implicitly assumed there are two additional hydrogen atoms bonded to it. Elements other than B, C, N, O, P, S, F, Cl, Br and I must be written inside a square bracket, for example “[Si]” for silicon.

For a given molecule there will exist a large number of equally valid ways of writing the molecule. For example ethanol can be written as either “CCO”, “OCC” or “C(C)O”. Double bonds are written with a “=” and triple bonds with a “#”. Charges, isotopes, chirality and non-implicit hydrogen counts are written within a square bracket for the atom in question, for example [15NH4+] for an isotopically labelled nitrogen ammonium cation. Canonicalisation techniques exist to generate a unique SMILES for a given structure. These differ slightly in implementation between different program vendors which means they are only canonical for a given version of a program.

SMILES is commonly used due to its compact nature. It can easily be inserted into a spreadsheet column while non line-format chemical representations cannot. There is no standard method of representing a polymer repeat unit using SMILES. A variety of different non-standard methods have been used by different groups including using a ring with index 0 to represent the ends of the repeat unit [11], and using the “\$” and “_” characters to mark the joins between repeat units[12] (see Chapter 2). The lack of directionality in the index 0 method means it is unsuitable for use with copolymers as there is no way to specify which way round the different repeat units combine.

1.2.3 XML

XML (eXtensible Markup Language)[13] is a markup language designed to be extensible and human-readable. XML consists of a tree structure where each

node can have any number of children, but must have exactly one parent. The document must contain a root node to be the parent of the top-level nodes.

There are three main types of nodes, *elements*, *attributes* and *text nodes*. Only elements are allowed to have child nodes other than text. An example XML document is shown below:

```
1 <?xml version="1.0" ?>
2 <bookstore>
3   <book>
4     <title lang="en">Prediction of Polymer Properties</title>
5     <author>Jozef Bicerano</author>
6     <year>2002</year>
7     <price currency="GBP">60</price>
8   </book>
9 </bookstore>
```

In this XML document the root element is *bookstore*. Element names are enclosed in angle brackets. The element continues until it is closed with a `</ElementName>` tag. Elements which do not have any children can be closed with a `"/"` before the `">"` like so: `<emptyElement/>`

One of the advantages of XML is that it is extensible. This means that any new elements can be added to an XML document without breaking the existing semantics. This allows new information to be encoded into XML without breaking other software, or conflicting with other additions from other sources. We could extend the syntax above to add an `<isbn>` element. This new element would simply be ignored by older software that could not make use of the extension, rather than being completely incompatible.

```
1 <?xml version="1.0" ?>
2 <bookstore>
3   <book>
4     <title lang="en">Prediction of Polymer Properties</title>
5     <author>Jozef Bicerano</author>
6     <year>2002</year>
7     <price currency="GBP">60</price>
8     <isbn>0-8247-0821-0</isbn>
9   </book>
10 </bookstore>
```

XML is also query-able using the XQuery language [14]. XQuery allows complex queries to be run over a set of data held in XML files. These queries

can be used to transform XML data into HTML for display purposes, or extract specific pieces of information out of an XML database. For example, to extract all the titles in English from a series of documents, the XQuery “//title[@lang="en]” will select them. This simple XQuery is also an XPath expression, but more complicated XQueries can be written using the FLWOR (For, Let, Where, Order by, Return) syntax. An example, if we wanted to extract a list of all the titles with a price in pounds, ordered by price we could use the XQuery:

```
1 for $x in /bookstore/book
2 where $x/price/@currency eq "GBP"
3 order by $x/price
4 return $x/title
```

Here the **for** binds **\$x** to a set of all <book> elements which are children of <bookstore>. The **where** selects only entries which match the condition that the **currency** attribute has value “GBP”. The **order** statement then sorts the list by price order and the **return** statement indicates that the <title> elements should be returned. This XQuery therefore returns all the <title> elements in price order if the currency is in sterling. XQuery is used in Chapter 5.

There exist a range of tools that use XML documents as processing instructions to modify other XML documents. An example of this is XSLT (Extensible Stylesheet Language Transformations). XSLT is commonly used to transform an XML document into an HTML document for displaying to users on the web, however XSLT can be used to transform an XSLT document (since XSLT are written in XML). This allows for self-modifying documents. This concept is used in PML (Polymer markup language), which is described in Chapter 3.

CML

Chemical Markup Language (CML)[15, 16, 17, 18] is an eXtensible Markup Language (XML) file format. CML takes advantage of the extensible nature of XML[13] to produce an extensible chemical data format. The core CML format consists of <molecule> elements which contain information about

```

1  <?xml version="1.0"?>
2  <molecule xmlns="http://www.xml-cml.org/schema" id="Ethanol">
3    <formula inline="CCO" convention="SMILES"/>
4    <atomArray>
5      <atom id="a1" elementType="C"/>
6      <atom id="a2" elementType="C"/>
7      <atom id="a3" elementType="O"/>
8      <atom id="a4" elementType="H"/>
9      <atom id="a5" elementType="H"/>
10     <atom id="a6" elementType="H"/>
11     <atom id="a7" elementType="H"/>
12     <atom id="a8" elementType="H"/>
13     <atom id="a9" elementType="H"/>
14   </atomArray>
15   <bondArray>
16     <bond atomRefs2="a1 a2" order="1"/>
17     <bond atomRefs2="a2 a3" order="1"/>
18     <bond atomRefs2="a1 a4" order="1"/>
19     <bond atomRefs2="a1 a5" order="1"/>
20     <bond atomRefs2="a1 a6" order="1"/>
21     <bond atomRefs2="a2 a7" order="1"/>
22     <bond atomRefs2="a2 a8" order="1"/>
23     <bond atomRefs2="a3 a9" order="1"/>
24   </bondArray>
25 </molecule>

```

Table 1.2: CML listings for an ethanol molecule with explicit hydrogens.

that molecule. This can be a list of atoms and bonds, a name, a SMILES string, an identifier from a registry service such as a CAS (Chemical Abstracts Service) number or data about the molecule. As the format is XML any additional data can be stored in the CML file as non-CML XML if required. Table 1.2 contains an example CML listing for ethanol. CML is normally only used to depict small molecules, but has been extended into Polymer Markup-Language (PML) to represent polymers.

1.3 Computer representation of polymers

The previous formats are used to represent small molecules, but representing a polymer requires some additional information.

1.3.1 MDL mol for repeat units

The mol format has provided some capacity for representing polymer repeat units. In Table 1.3 the listing for a mol file representing the repeat unit for

```

1
2   Marvin   08181013002D
3
4   5   4   0   0   0   0           999 V2000
5   -5.8677   6.2254   0.0000 C   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
6   -5.1680   6.6626   0.0000 H   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
7   -5.8389   7.0499   0.0000 H   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
8   -5.4552   5.5109   0.0000 *   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
9   -6.6927   6.2254   0.0000 *   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
10  1   2   1   0   0   0   0
11  1   3   1   0   0   0   0
12  1   4   1   0   0   0   0
13  1   5   1   0   0   0   0
14 M   STY   1   1   SRU
15 M   SCN   1   1   HT
16 M   SAL   1   3   1   2   3
17 M   SDI   1   4   -6.2802   7.4624   -6.2802   5.8129
18 M   SDI   1   4   -4.7555   5.8129   -4.7555   7.4624
19 M   SBL   1   2   3   4
20 M   SMT   1   n
21 M   END

```

Table 1.3: mol file listings for poly(ethene) repeat unit.

poly(ethene) is shown. Repeat units in a mol file use the S-group data block. Here the atoms labelled with a “*” represent atoms in the neighbouring repeat units. Line 14 starts with STY (S-group Type) and specifies that there is 1 S-group (1) and this S-group’s type (1) is a Standard Repeat Unit (SRU). Other options include monomer(MON), copolymer (COP) and any-polymer(ANY).

Line 15 starts with SCN, for S-group Connectivity, then specifies that the repeat units are to be joined head-to-tail (HT). Other allowed options are head-to-head (HH) and either/unknown (EU). Line 16 contains the S-group atom list (SAL). This contains the S-group number (1) followed by the number of atoms in the group (3) and then the numbers of the atoms which are in the group (1, 2, 3).

Lines 17 and 18 contain S-group Display Information (SDI). This is simply x and y co-ordinates for the square brackets of the repeat unit. This is used to represent how the molecule was drawn in a software package rather than information about the molecule itself. Line 19 contains the S-group bond list (SBL). This is the S-group number (1), followed by the number of bonds (2) followed by the bond numbers (3, 4). Line 20 contains the symbol to be used after the S-group, in this case an “n”.

In a copolymer two different S-groups would be used, and the S-group type would be specified as (COP). A block, alternating or random copolymer can be specified in an S-group subgroup type line (SST). There is no way to specify a statistical copolymer in the mol format.

1.3.2 Polymer Markup Language (PML)

PML[19] is discussed in detail in the Chapter 3. As well as being able to represent a simple repeat unit, PML is capable of representing statistical copolymers and dendrimers which other formats are unable to represent. PML is an XML format which can represent a polymer sample, that is an ensemble of macromolecules, rather than a single macromolecule. Individual CML macromolecules can be created from the PML document allowing a sample to be enumerated.

1.3.3 PoLyInfo Formula

This format is used by the PoLyInfo database to hold their polymer data. There is only a small amount of documentation available on the format, which is discussed in Section 4.6, along with a tool that has been created, in the course of this work, to convert a PoLyInfo formula into PML.

A PoLyInfo Formula is used to represent a repeat unit. It cannot represent a statistical copolymer

Chapter 2

Application of Polymer Informatics to Polymer Selection for Concentrated Cleaning

2.1 Aim

Unilever has a business need to continually improve the cost-effectiveness of its cleaning products. Innovating with polymeric ingredients offers opportunities to do this. In this Chapter, a study into using machine learning techniques on polymer data is discussed. This data was gathered in a controlled manner with standard procedures by one organisation. In order to test the applicability of using informatics techniques on polymers, this dataset was chosen for an initial study. Since the data was all gathered by one organisation, the consistency should be high, and the chance of seeing a signal reasonable.

The aim was to see to what extent could the cleaning efficacy data produced by Unilever be used to build predictive models for polymer effectiveness on cleaning. As the results of this study are a trade-secret some details have had to be withheld.

2.2 Background

An important ambition in home care is to produce concentrated cleaning products. Since a concentrated product contains less water, it requires less packaging and can be transported for a lower cost and with lower carbon footprint. Unilever have achieved a three-fold increase in concentration with current technology. Every effort is being made to realise benefits from further concentration, but new technologies are needed for this.

The aim of the project being supported by the work reported here is to investigate the design of novel polymer technologies for this purpose. Polymers are interesting in this context, and product innovation in general, not only because of opportunities for superior performance, but because the introduction of novel polymers into the market comes with much lower risk of harm than introducing novel small molecules. This is reflected in the recent REACH (Registration, Evaluation, Authorisation and Restriction of Chemical substances) European directive, which reduces the burden of safety testing on polymers made from most commercially exploited monomers compared to small, discrete molecules.

2.3 Predicting Properties

When designing small molecules for the drug industry, the standard approach is one of structure-property relationship. A high-throughput screen of a range of diverse molecules is performed, then models correlating the properties to the structure of the molecules are built. A more detailed screen can then be carried out around the chemical space of the “good” molecules. This process can be repeated to refine the chemical space that is being investigated.

This works well for small molecules, but there are a number of key differences between these and polymers. For small molecules such as ethanol you assume that all the molecules in your sample are $\text{CH}_3\text{CH}_2\text{OH}$. This approximation works because the differences between molecules will either be due to isotopes, and so have negligible effect on the chemistry, or be reversible reactions involving water. A sample of a polymer, however, is an ensemble of

different macromolecules, which vary in terms of molecular weight, degree of branching, tacticity, distribution of co-polymers and initiating or terminating groups. These differences can result in large differences in properties, and so properties cannot be estimated solely on the basis of the monomers used to synthesise the polymer. The same monomer is used to make polyethene bags as high-density rubbish bins. The differences in physical properties are due to the branching and molecular weight differences.

Because the properties of polymers depend on more than the structure of the monomers used to make it a way of representing the structure of the polymer is required.

2.4 Polymers

The polymers used in the high-throughput screening come from a variety of sources. Some come from a range of suppliers and are polymers that are essentially taken “off-the-shelf”, others have been synthesised previously by Unilever employees, the remaining forty novel polymers have been specifically synthesised for this project by Sue Rogers.

The data we have about the structure of these polymers are the structure of the repeat unit of the monomers used in the synthesis, the relative reactivity ratios of these, the terminating groups, the average molecular weight and the polydispersity index. The polydispersity index was not available for many of the polymers, however.

The repeat unit of a monomer is the chemical structure that will result from the addition of a unit of that monomer into a growing macromolecule. It is represented as a non-standard SMILES string using a \$ character to label the atom that should be bonded to the previous repeat unit and a _ character to label the atom which should be bonded to the next repeat unit. This method of representation was devised by Ian Stott[12]. Initiating groups have just a _ label, terminating groups only a \$ label. The repeat unit for polyethene capped with hydrogens would therefore be represented as “C\$C_” with a “[H]_” initiating group and a “[H]\$” terminating group.

The relative reactivity ratios were represented in an n-by-n matrix where

n is the number of different repeat units in the polymer. The values in this matrix give the relative probability of a particular repeat unit given the previous repeat unit. This allows for any structures which can be described by a Markov chain to be represented. As the SMILES are directional the directionality of the polymer can be specified. If the repeat units can be added both head-to-head and head-to-tail then both directions should be listed as separate repeat units. This representation cannot represent changes in concentration of monomer over the course of a reaction.

Some of the polymers have side groups represented by an R in the SMILES string. These must first be dereferenced and the structure of the R group added to the repeat unit in question. Some of these side groups themselves consist of a repeated repeat unit, which must first be polymerised into a larger fragment before being added to the main chain.

A Pipeline Pilot[20] protocol was constructed which built a representative macromolecule of the polymer from this data. Side-chains are constructed first and added to their respective main chain repeat units. Then the repeat units are stochastically added according to the reactivity ratios in the probability matrix. The repeat length of the macromolecule depends on the average molecular weight of the polymer to be modelled, but since the generation is a stochastic process it will vary to some extent.

A problem was encountered with highly branched polymers due to the limitations of the SMILES format, which allow only 99 unclosed bonds at one time. This necessitated holding the structures in an invalid SMILES format briefly, before using Pipeline Pilot to add the required bonds to the finished molecule.

The cross-linked polymers are not currently built due to implementation difficulties with the current method of building polymers. It is possible to build cross-linked polymer structures in theory, but since a cross-linked polymer will have a very large, often referred to as "infinite", molecular weight it is not practical to represent them atomistically in a computer.

2.5 Descriptors

In order to use machine learning methods on a dataset involving molecules it is necessary to represent the molecules in some way that enables the computer to compare different molecules. Two molecules that differ only slightly can be regarded as “similar” while two molecules with a vastly different structure are not similar. Much of chemoinformatics is based on this *similar property principle*.

A large body of work exists for the prediction of properties of small molecules using descriptors. These are across diverse areas such as boiling point prediction[21], solubility prediction[22], NMR shift prediction[23] and biological activity[24]. Typically a range of descriptors are calculated from the chemical structure of the molecules in question. A relationship between these descriptors and the desired property is developed using either some form of regression or machine learning methods. To make a prediction, the descriptors for the molecule of interest are calculated and fed into the model to produce a predicted value.

Work has been done by Bicerano[25], at the Dow Chemical Company, implementing these concepts for polymers. Bicerano calculates a range of polymer properties by calculating a range of descriptors based on the repeat unit of the polymer and using these as variables in an equation to give a predicted value. This approach does not allow for variation in properties between samples of the same polymer as all samples of a given polymer will share the same repeat unit.

There are thousands of descriptors which can be calculated, these generate a number based on the molecular structure of the molecule. For example, the Weiner index[26] is the sum of all distances between all pairs of atoms in the molecule. It therefore increases exponentially with increasing molecular weight of polymer. Other descriptors scale with molecular weight in a different fashion. The vast majority of the descriptors scale linearly with molecular weight such as the Zagreb index [27] while a few tend to some limit such as the Complementary Information Content (CIC) descriptor [28]. This scaling behaviour can be seen in Figure 2.1.

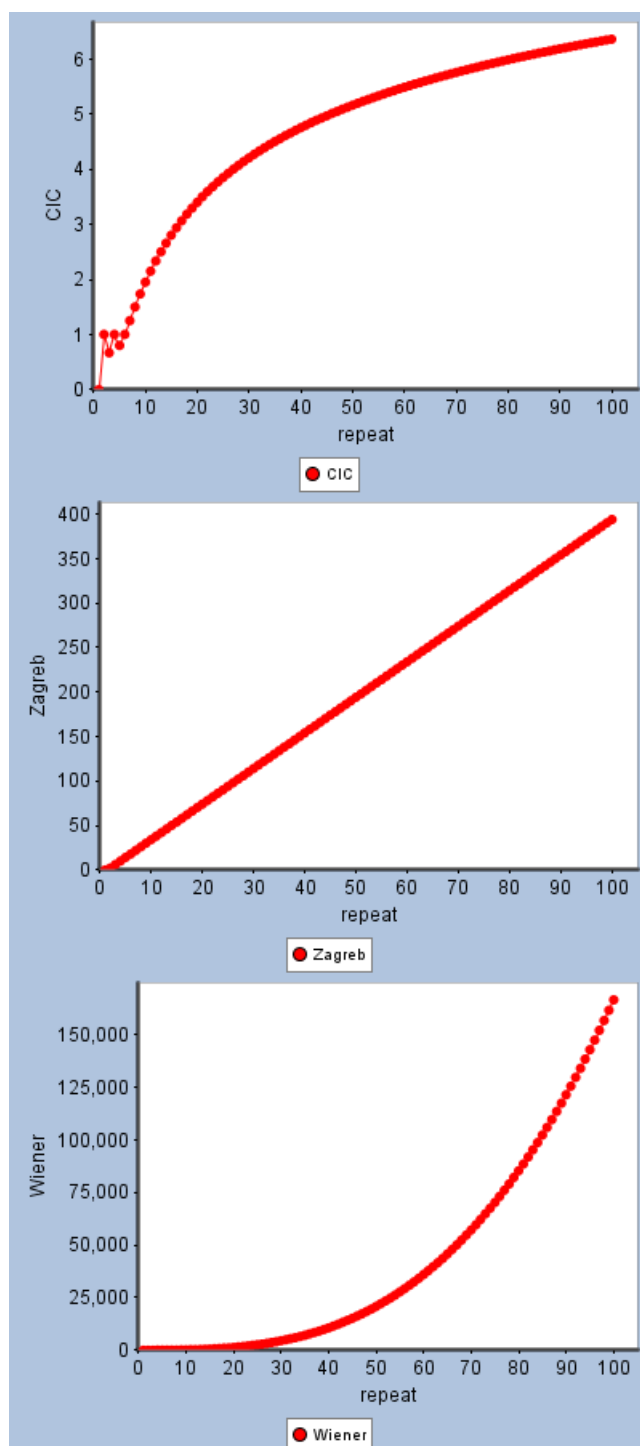


Figure 2.1: Diagram to show how descriptors scale with degree of polymerisation for a straight-chain macromolecule.

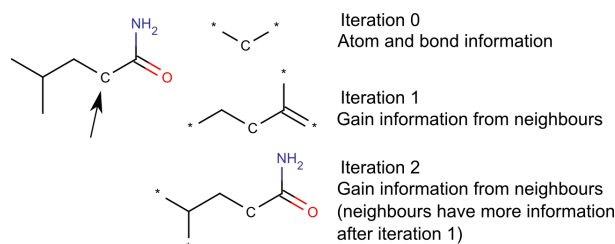


Figure 2.2: The process of SciTegic fingerprint generation. Initially each atom is assigned a number based on its element and number of bonds. This contains the information shown for iteration 0. During each iteration, the numbers of each neighbouring atoms are appended to the number of the original atom and hashed to produce a new number. This adds the information from the previous iteration of each of an atom's neighbours. In iteration 2, the information from an atom's neighbours in iteration 1 is added. The process for the carbon atom marked with the arrow is shown.

For this reason the set of descriptors are calculated and used as both the raw descriptor, and a scaled descriptor divided by the molecular weight of the macromolecule. This approach means that the learning methods can use the descriptors or scaled descriptors as they see fit and discard the non-informative ones.

The descriptors that were used were the Balaban[29], Wiener[26], Zagreb[27], Chi Index (up to degree 3)[30], Kappa Shape Index (up to degree 3)[31], Phi Molecular Flexibility, InfoContent descriptors[28] and Subgraph Count Index. They were all calculated from within Pipeline Pilot using the appropriate SciTegic components. Molecules over around 17,000 molecular weight were too large to have these descriptors calculated and had to have the descriptors for a 17,000 molecular weight molecule calculated instead.

As well as descriptors, fingerprints were also used. The fingerprints used were the Scitegic FCFC fingerprints. These use a Morgan[32] style algorithm to encode atom environments up to six atoms in radius iteratively. The count of the number of times each environment occurs as well as the counts per molecular weight were calculated and used. This process is shown in Figure 2.2. This allows specific chemical groups to be included in the model. The fingerprints use hashes but the SciTegic calculator records the pre-hashed

SMILES string corresponding to each fragment so it is possible to match the fingerprint IDs to the relevant molecular fragment later.

As well as the descriptors and fingerprints describing the polymer in question, there is additional data about the conditions under which the cleaning test was conducted. This ranges from the fabric type, water hardness and amount of polymer added to the drying time and number of wash cycles. The factors that were the same for all of the dataset were removed, since they add no information to the model.

2.6 Machine Learning Methods

There are many different machine learning algorithms available. The data set in question contains both categorical and numerical data. This means that not all learning methods are suitable since some can only deal with one type of data, e.g. a regression model cannot deal with categorical variables. In this study the Scitegic Bayesian Inference model, the Waikato Environment for Knowledge Analysis (WEKA)[33] Classification And Regression Tree (CART) model and the WEKA Random Forest[34] model were used. These methods are also good at dealing with large numbers of non-informative properties, which can confuse other machine learning algorithms.

The Bayesian method assumes that each property is independent of all the other properties. In a binary classification problem the data is divided into two classes, such as "good" and "bad" depending on if the desired property is present or not. The properties are divided into a number of bins and the ratio of the number of times "good" data has a value within that bin rather than "bad" data is used to estimate a probability that having a property value within that bin means the data is more likely to be "good" or "bad". The sum across all the properties is then used to estimate the chance that the data value will be "good" or "bad". Depending on where the cut-off value is taken the trade-off between recall and precision can be adjusted. Having too high a cut-off will reduce the recall of "good" data values, but reduce the rate of false-positives improving the precision. Too low a cut-off will have very good recall, but an unacceptably high false-positive rate and so a very

low precision.

When assessing the reliability of a model the cross-validated results are important. This is when the model is tested on data that it has not been trained on. A random split of the data into two halves was used to perform a two-fold cross validation. This is a harsher test of the model than a 3-fold or higher as the proportion of training data is lower. Because the modelling method is deterministic the same model will be built from the same data set every time. With a random split for the cross-validation, however, a different model will be built and tested on a different test set each time. For this reason the cross-validated scores were calculated multiple times to give a more reliable indication of model robustness.

The CART method is also deterministic and works by splitting the data set based on one property in order to maximise the information gain, then splitting both of these leaves again until no more splits can be achieved. Finally pruning with a 0.25 confidence threshold takes place to remove unnecessary nodes in order to avoid overfitting the data. The model is again judged by a two-fold cross validation.

The random forest learning algorithm works by generating a large number of decision trees. Each tree only considers a bootstrap sample of the data. (This is where for a set of N data points N individual data points are chosen randomly with replacement. This gives some duplicate data points and some absent data points.) At each split in the tree only a small random subset of the total number of properties is used. This ensures that the trees will not be too similar. No pruning takes place on these trees. After constructing a large number of these decision trees the model makes predictions by a “vote” of the individual trees weighted by their estimated error from the data samples they were not trained on (out-of-bag error.)

Because a random forest is generated randomly, repeating the model building process will result in a different model and thus different results. This means that the model score as well as the cross-validated score will differ with repetitions for a random forest.

2.7 Data

The data are divided initially into two categories: the larger set of “off-the-shelf” polymers and the smaller set of custom built polymers. Each of these data sets is further subdivided into different soil classes (A, B & C) depending on the type of soiling used in the test. There are two soil types within each class.

The first larger dataset comprises of 5142 A soil datapoints, 2572 B soil datapoints and 2572 C soil datapoints, with 184 different polymers that could be built. This dataset contains a very diverse and large number of polymers. The results of the cleaning experiment are given as a “good”, “bad” or “no significant result”. This dataset only has four of the five soil types, missing out one B soil.

The second dataset contains 1824 A soil, 1824 B soil and 912 C soil datapoints, with 40 different polymers. However there were reported experimental problems when the water hardness was greater than four French Hardness units (40 milligrams of calcium carbonate per litre of water); the polymer was precipitating out of solution. For this reason the data with a French Hardness of 4 or less was analysed separately from the rest. The significance of the cleaning experiments is given one of five scores: + +, +, 0, -, - -. These refer to very significantly good, significantly good, no significance, significantly bad and very significantly bad. For the analysis of this data the + and ++ were counted as belonging to the active class, with the remaining being in the inactive class.

Both datasets consist of experimental data where each polymer has been tested under a range of conditions. These conditions include a range of properties such as the type and amount of base added to the solution, other additives and their amounts, solution pH and pCa levels, wash time, number of wash cycles, surfactant type and quantity, and the amount of polymer.

2.8 Results and Analysis

Models were generated using a Pipeline Pilot protocol. The experimental data was merged with the descriptors and fingerprints from the relevant polymer to provide an enriched data set. Predictions were made and the results analysed using a ROC plot.

A Receiver Operating Characteristic (ROC) plot is the plot of the true positive rate against the proportion of false positives. The Area Under the Curve (AUC) of the ROC plot is equal to the probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one. An AUC of 0.5 is therefore the same as random chance and a score of 1 is perfect prediction. The ROC AUC score for the training set was computed, as well as the 2-fold cross-validated ROC AUC score with a random split. This cross-validated score provides a indication of how reliable the model is for predicting new data, while the non-cross-validated score only indicates how well the model describes the training data. If the cross-validates and non-cross-validates scores are similar, then the model is probably robust. If they are dissimilar then the model is probably overfitted to the training data and will not give reliable results outside the training set.

2.8.1 Soil Class A

The results of Soil Class A for a water hardness of 4 or less French Hardness units are given in the Table 2.1. The “XV2 AUC” column shows the cross-validated ROC AUC scores.

The models for the A soil class are extremely good, with cross-validated ROC AUC scores of around 0.9. The ROC plot for the CART model is shown in Figure 2.3.

The y-axis gives the true positive rate, which is the fraction of true positives identified while the x-axis gives the false positive rate, which is the proportion of negatives identified as positives. If the data is ordered according to how confident the model is that the datapoint is positive, and then descended from the top of the list you will traverse the line from (0,0) to (1,1). The area under the curve then gives the probability that the model

Modelling Method A Soil	ROC AUC	XV2 AUC	Mean XV2
Scitegic Bayesian Inference	0.891	0.856	0.860
Scitegic Bayesian Inference	0.891	0.869	
Scitegic Bayesian Inference	0.891	0.853	
CART	0.967	0.885	0.906
CART	0.967	0.908	
CART	0.967	0.926	
Random Forest	0.991	0.909	0.905
Random Forest	0.991	0.902	
Random Forest	0.992	0.905	

Table 2.1: Table of prediction results for A soil class.

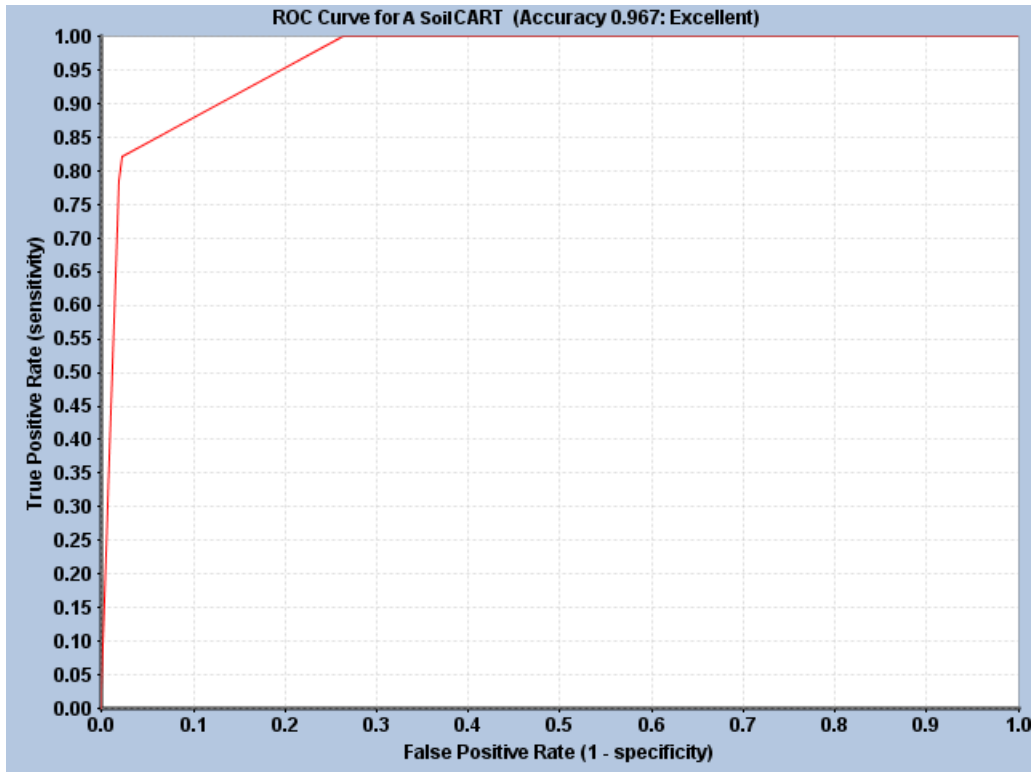


Figure 2.3: ROC plot for A soil class CART model.

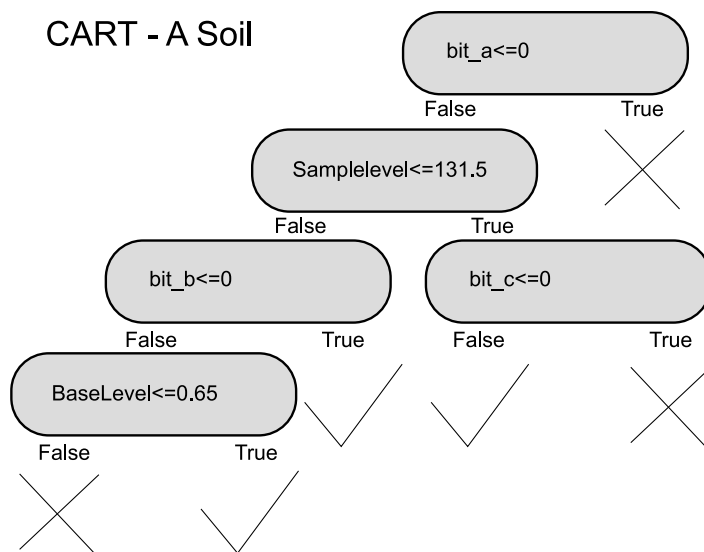


Figure 2.4: Depiction of CART tree for A soil class. The cross symbols mean the polymer is predicted to have not-good cleaning efficacy, while the tick symbol means the polymer is predicted to have good cleaning efficacy. The properties bit_a, bit_b and bit_c correspond to specific substructures found in the macromolecules.

can distinguish between any two randomly chosen positive and negative data points[35], in this case 0.967.

The CART model that gives this prediction is reasonably simple and is shown in Figure 2.4. If a statement is true, then the tree continues down the right hand side, if false then the left hand side, until it reaches a cross or tick which represent not-good and good cleaning efficacy respectively.

Each bit corresponds to a different fingerprint which represents a different substructure. The exact nature of the sub-structures is a trade secret, but they consist of between 10 and 19 atoms.

The property bit_a corresponds to a 13 atom substructure which corresponds to the inclusion of a specific monomer. The CART model predicts this substructure is beneficial for a polymer to include, since presence of the substructure produces a value greater than 0, and values less than or equal to zero are predicted “bad”. Since negative values are impossible this is in practise just checking for the presence of a substructure corresponding to bit_a. The test “bit_a ≤ 0” is false if the substructure is present, and true if it is

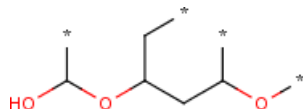


Figure 2.5: Diagrammatic representation bit_d, which was not used in the model but serves as a demonstration substructure. This substructure has the SMILES string [*]CC(OC(O)[*])CC(O[*])[*]

absent. It would be more intuitive to test “bit_a>0” instead and invert the tree, but both are equivalent as far as the CART model is concerned.

The property bit_b corresponds to a 19 atom substructure. This split is only on a limited number of data points however, and is probably an artefact of the dataset used, since it only occurs in one of the polymers. The absence of this substructure in the polymer makes the CART model predict “good” cleaning performance.

Finally bit_c corresponds to a 10 atom substructure which was predicted to be beneficial. The model therefore predicts that a polymer will have good cleaning efficacy if it contains the substructure fragment represented by bit_a, and either bit_c or not containing bit_b while having a sample level greater than 131.5.

An example substructure, bit_d, which was not used in the model is shown in Figure 2.5. This substructure has the SMILES:

```
1  [*]CC(OC(O)[*])CC(O[*])[*]
```

Any molecule which includes this substructure will have a bit_d value equal to the number of times this substructure occurred. These substructures were generated using the SciTegic FCFC fingerprints.

It is interesting that the CART model is so simple. This is because there was a strong signal to be found. The other descriptors not used in the CART model might still contain some information, but not sufficient to significantly improve the model.

The random forest model performs similarly to the CART model, but with a greater variance. This is to be expected as a random forest is essentially a collection of individual CART models, each trained on a sub-set of the data. Interpreting a random forest is a much more difficult task than interpreting

Modelling Method B Soil	ROC AUC	XV2 AUC	Mean XV2
Scitegic Bayesian Inference	0.861	0.655	0.685
Scitegic Bayesian Inference	0.861	0.734	
Scitegic Bayesian Inference	0.861	0.665	
CART	0.500	0.436	0.428
CART	0.500	0.410	
CART	0.500	0.436	
Random Forest	0.973	0.602	0.659
Random Forest	0.975	0.660	
Random Forest	0.969	0.714	

Table 2.2: Table of prediction results for B soil class.

a CART model, since it consists of a large number of differently weighted trees.

2.8.2 Soil Class B

The results for the B soil class with a water hardness of 4 or less French Hardness units is given in Table 2.2.

The CART model is no better than random, with an AUC of 0.5 and so holds no information. The random forest models have a high score on the training set, but they drop considerably on the cross-validated AUC, the Bayesian model performs slightly better but the cross-validated AUC are still below 0.7.

2.8.3 Soil Class C

The C soil class data contained no “good” results with a water hardness of 4 French Hardness units so all water hardness data was considered. The results are shown in Table 2.3.

This is even worse than for the B soil class, and is not much better than random chance. The fact that there were no positive hits for a hardness of 4 and the data with a hardness of more than 4 is suspect means that this is to be expected.

Modelling Method C Soil	ROC AUC	XV2 AUC	Mean XV2
Scitegic Bayesian Inference	0.798	0.654	0.636
Scitegic Bayesian Inference	0.798	0.594	
Scitegic Bayesian Inference	0.798	0.661	
CART	0.660	0.528	0.511
CART	0.660	0.608	
CART	0.660	0.398	
Random Forest	0.985	0.594	0.575
Random Forest	0.981	0.495	
Random Forest	0.980	0.634	

Table 2.3: Table of prediction results for C soil class.

2.9 Conclusions

The results for the A soil class models are very good, with a ROC AUC score above 0.9. The most important factors were the presence of a structural motif present in the repeat unit of one of the monomers, and a polymer amount above 131.5. In the case of the C soil class there were no good results for experiments with a French Hardness of 4 or less, and since the results for higher hardness levels have problems with the polymer precipitating out of solution it is not unexpected that there is no useful information in the models. The B soil class models show some ability to predict, except the CART model, which performed no better than random chance. The random forest and Bayesian models still performed poorly. Whether this is due to the methods used to analyse the data or the data itself is difficult to answer definitively.

The macromolecules used here were generated to be representative of the polymer. An extension of this approach would involve the generation of an ensemble of macromolecules. This would allow a more accurate representation of the polymers which might provide a better model.

In Chapter 3 the use of PML to represent polymers is discussed. PML can be used to represent an ensemble of macromolecules. In Chapter 7 the data in the PoLyInfo database is used for a regression model using a similar approach to the work in this Chapter. The data from the PoLyInfo database is heterogeneous as it is from a great variety of sources and so not as controlled

as the data from Unilever.

Chapter 3

Polymer Markup Language

3.1 Introduction

In Section 1.3 in Chapter 1 various formats for storing a polymer's structure were described. These formats allow the repeat unit of a polymer to be stored electronically. However since a polymer is an ensemble of macromolecules these formats do not represent a sample of the polymer, only the general archetype. In order to represent an ensemble of macromolecules it is necessary to use a format which allows for the enumeration of a representative number of macromolecules. For this purpose Polymer Markup Language(PML) can be used. PML can be used to specify a repeat unit representation of a polymer, but in addition it is possible to specify a polymer in terms of an ensemble of macromolecules. When a polymer has been specified in such a manor, an arbitrary number of explicit individual macromolecules in CML can be produced from the PML document, representing the variation in structure. This allows a PML document to represent a sample of a polymer.

In this section the syntax and usage of PML is explained with examples; starting with straightforward examples and proceeding to an extension of the PML language which allows the representation of more complicated polymer samples such as an ethylene/propylene copolymer where the ratio of the two monomers varies with the molecular weight of the macromolecule in the

sample. It is not possible to fully represent such a sample in any other format.

3.2 Syntax

When this Thesis was started there was no PML. The core PML language was developed by Peter Murray-Rust and Nico Adams over 2006-2008. Development of use cases, tests and build infrastructure was produced in the course of this Thesis. PML is an XML dialect that is heavily related to the CML XML dialect (described in 1.2.3), but has been designed specifically for describing polymers. Since it is not always possible to accurately represent a polymer with a traditional connection table (for example, owing to variable molecular weight or degree of branching) a different approach is needed. PML can be used to describe either a single molecule, or to represent an ensemble of molecules.

3.2.1 Core Elements

The elements which make up PML are[19]:

<fragment> This element represents a part of the polymer. It can either contain another **<fragment>**, a **<molecule>** element, a reference to another fragment or a **<fragmentList>**.

<join> This element is used to specify how the **<fragment>** elements are joined together. It contains an **atomRefs2** attribute that specifies which atoms are to be joined together.

<fragmentList> A **<fragmentList>** element is a container for a number of **<fragment>** elements.

3.2.2 CML

PML documents also include CML elements and attributes. The most important of these are:

```

1 <?xml version="1.0"?>
2 <map xmlns="http://www.xml-cml.org/schema">
3   <link convention="cml:relativeUrl" from="http://www.xml-cml.org/mols/geom1"
4     role="cml:moleculeList" to="./geom1"/>
5   <link convention="cml:relativeUrl" from="http://www.xml-cml.org/mols/polyinfo"
6     role="cml:moleculeList" to="./polyinfo"/>
7 </map>

```

Table 3.1: Catalog.xml listing which allows URIs to be resolved to physical files.

<molecule> This element is used to contain the connection table of a molecule.

id This attribute is used to define a reference to an XML element. Elsewhere a copy of the element with this attribute can be invoked by using the **ref** attribute.

ref This attribute is used on an element to specify that this XML element should be replaced with a copy of the referenced element.

3.2.3 ref and id attributes

PML utilizes a **ref** and **id** attribute system for referencing and de-referencing XML elements. The **id** attribute is used to label a **<fragment>** or **<molecule>** element such that it can be referenced elsewhere. The **ref** attribute is then used to de-reference the previously declared **<fragment>**. In this way a **<fragment id="f1">** can be referenced elsewhere with a **<fragment ref="f1">**. The same syntax applies to **<molecule>** elements. The scope of these **id** attributes is limited to the same document if no namespace prefix is used. If a namespace prefix is used, then the namespace URI will specify where the file can be found with the required **id** attribute in a catalog. An example catalog file is shown in Table 3.1.

3.2.4 <molecule>

A CML **<molecule>** element contains a connection table for a molecule. Since these files can be verbose it is convenient to store the **<molecule>** element in a separate file with an **id** attribute and reference it with a **ref**


```

1  <?xml version="1.0"?>
2  <molecule xmlns="http://www.xml-cml.org/schema" id="eo">
3    <atomArray>
4      <atom id="r1" elementType="R" x3="1.426000" y3="0.543000" z3="0.175000"/>
5      <atom id="a2" elementType="C" x3="0.703000" y3="-0.593000" z3="-0.302000"/>
6      <atom id="a3" elementType="C" x3="-0.703000" y3="-0.593000" z3="0.302000">
7        <label dictRef="cml:torsionEnd">r1</label>
8        <label dictRef="cml:torsionEnd">r2</label>
9      </atom>
10     <atom id="a4" elementType="O" x3="-1.426000" y3="0.543000" z3="-0.174000"/>
11     <atom id="a6" elementType="H" x3="0.631000" y3="-0.548000" z3="-1.389000"/>
12     <atom id="a7" elementType="H" x3="1.224000" y3="-1.505000" z3="-0.010000"/>
13     <atom id="a8" elementType="H" x3="-1.224000" y3="-1.505000" z3="0.009000"/>
14     <atom id="a9" elementType="H" x3="-0.631000" y3="-0.549000" z3="1.389000"/>
15     <atom id="r2" elementType="R" x3="-2.304000" y3="0.505000" z3="0.230000"/>
16   </atomArray>
17   <bondArray>
18     <bond atomRefs2="r1 a2" order="1"/>
19     <bond atomRefs2="a2 a3" order="1"/>
20     <bond atomRefs2="a2 a6" order="1"/>
21     <bond atomRefs2="a2 a7" order="1"/>
22     <bond atomRefs2="a3 a4" order="1"/>
23     <bond atomRefs2="a3 a8" order="1"/>
24     <bond atomRefs2="a3 a9" order="1"/>
25     <bond atomRefs2="a4 r2" order="1"/>
26   </bondArray>
27 </molecule>

```

Table 3.2: Ethylene oxide CML file.

attribute when needed. An example of a CML file containing a repeat unit of ethylene oxide (eo) is shown in Table 3.2.

The connection table of the molecule consists of `<atom>` elements contained within an `<atomArray>` element, `<bond>` elements contained within a `<bondArray>` element. The `elementType` attributes on the `<atom>` elements specify the element of the atom. If the `elementType` attribute has the value of ‘R’ then the atom represents an extension of the molecule to currently undefined other atom(s). The `x3`, `y3` and `z3` attributes specify the 3D coordinates of the atoms. The `atomRefs2` attribute on the `<bond>` elements specify the `id` values of two `<atom>` elements. The `<bond>` element then represents a chemical bond between the two `<atom>` elements.

The parent `<molecule>` element has an `id` attribute with value “eo”. This enables the entire `<molecule>` element to be referenced with `<molecule ref="eo">` which will then be expanded into the complete CML listing shown at a later stage. The `<label>` elements will be discussed later in Section 3.4.2.

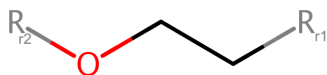


Figure 3.1: Representation of the `eo` molecule

A 2D drawing of this molecule can be seen in Figure 3.1. In Figure 3.1 the `id` attribute on the `<atom>` elements that represent R-groups have been shown. These `id` attributes are important as they are used to define where different `<molecule>` elements may join together.

Additional syntax will be introduced alongside a series of examples that illustrate progressively more complex features of PML.

3.2.5 Building CML from PML

A PML document can be converted into a CML document in order to provide a fully atomistic representation. We use an open-source Java toolkit called JUMBO[36] to perform this conversion. Details of how to perform this conversion are given in Section 3.7.

3.3 Example 1

In this example PML will be used to construct a larger molecule from three smaller molecule fragments. These smaller fragments are shown in Figure 3.2.

Syntax used in this example:

- `<fragment>` element
- `<join>` element

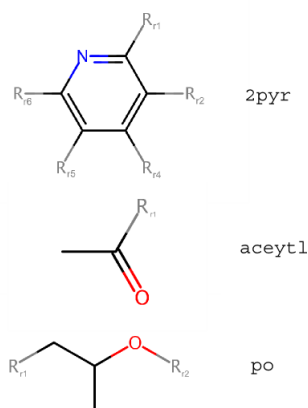


Figure 3.2: A diagram showing a representation of the three fragments used in Example 1.

- **countExpression** attribute

A common way to make use of the **ref** and **id** attributes is to include **<molecule>** elements from a catalogue of CML molecule files rather than writing the connection table directly into the PML file. An example PML file is shown in Table 3.3.

On line 5 from Table 3.3 the **ref** attribute is used to reference a molecule from other file. The contents of this attribute (**ref="g:2pyr"**) specify a URI to locate the molecule. This URI can be de-referenced using the namespace declaration in line 3 (**xmlns:g="http://www.xml-cml.org/mols/geom1"**), which binds the prefix *g* to that namespace in conjunction with the value **2pyr**. Combining a prefix and a local name in this way is called a QName (qualified name). Additional system-specific information is required to map the URI prefixes to the file address. JUMBO uses a catalog system to resolve these URIs. See Table 3.1 for the XML used. The 2pyr molecule is shown in Figure 3.2. This molecule could have been expanded into an explicit CML **<molecule>** rather than being referenced from another file.

```

1 <fragment convention="cml:PML-intermediate"
2   xmlns="http://www.xml-cml.org/schema"
3   xmlns:g="http://www.xml-cml.org/mols/geom1">
4   <fragment>
5     <molecule ref="g:2pyr" />
6   </fragment>
7   <join atomRefs2="r1 r2" moleculeRefs2="PREVIOUS NEXT" />
8   <fragment countExpression="*(2)">
9     <join atomRefs2="r1 r2" moleculeRefs2="PREVIOUS NEXT" />
10    <fragment>
11      <molecule ref="g:po" />
12    </fragment>
13  </fragment>
14  <join atomRefs2="r1 r1" moleculeRefs2="PREVIOUS NEXT"/>
15  <fragment>
16    <molecule ref="g:acetyl" />
17  </fragment>
18 </fragment>

```

Table 3.3: PML listing for Example 1.

3.3.1 <join> Elements

In order to join molecules together PML uses <join> elements. In Table 3.3 on line 7 is the first of these <join> elements. (Also shown below.)

```

7 <join atomRefs2="r1 r2" moleculeRefs2="PREVIOUS NEXT" />

```

The `atomRefs2="r1 r2"` attribute specifies that the `r1` atom of one molecule is to be joined to the `r2` atom of another. The `moleculeRefs2="PREVIOUS NEXT"` attribute specifies the two molecules being referred to. The value “PREVIOUS NEXT” states that the first molecule is the element before the <join> and the second is the one immediately afterwards. The other allowed value for `moleculeRefs2` is “PARENT CHILD”. “PARENT CHILD” specifies that the first molecule is an XML parent of the <join> element and the second molecule is an XML child of the <join> element. This can be used for attaching side-chains or branching.

3.3.2 countExpression

```

8 <fragment countExpression="*(2)">
9   <join atomRefs2="r1 r2" moleculeRefs2="PREVIOUS NEXT" />
10  <fragment>
11    <molecule ref="g:po" />
12  </fragment>
13 </fragment>

```

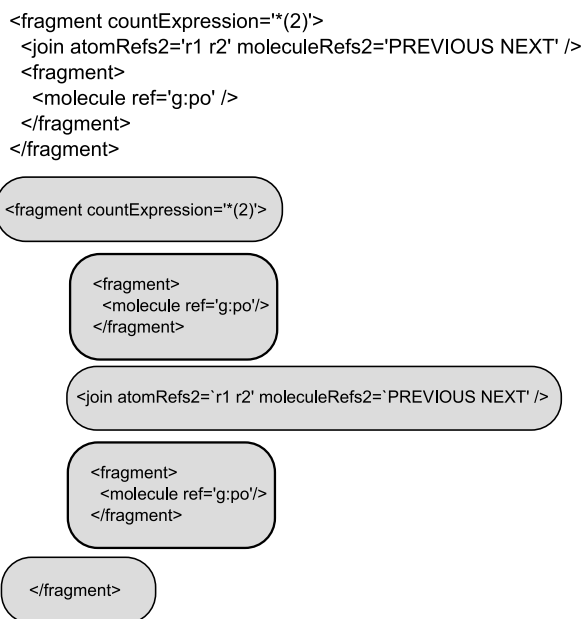


Figure 3.3: The expansion of a `countExpression` attribute. The `<fragment>` gets repeated twice, with the initial `<join>` after the `countExpression` placed in-between.

On line 8 a `<fragment>` element with a `countExpression` attribute is declared. This attribute is used to specify a repeated fragment. The syntax for this is a `<fragment>` with a `countExpression="*(n)"` attribute where n is the number of times the fragment is to be repeated. The repeated `<fragment>` is required to have a `<join>` element as its first child. This initial `<join>` is used to join together the repeated fragments. This is shown diagrammatically in Figure 3.3.

3.3.3 Complete Example

Returning to the complete text of the example in Table 3.3 we can now construct the whole molecule. Representations for the individual molecule fragments are shown in Figure 3.2. A diagram showing how the molecule is constructed as well as the completed molecule is shown in Figure 3.4.

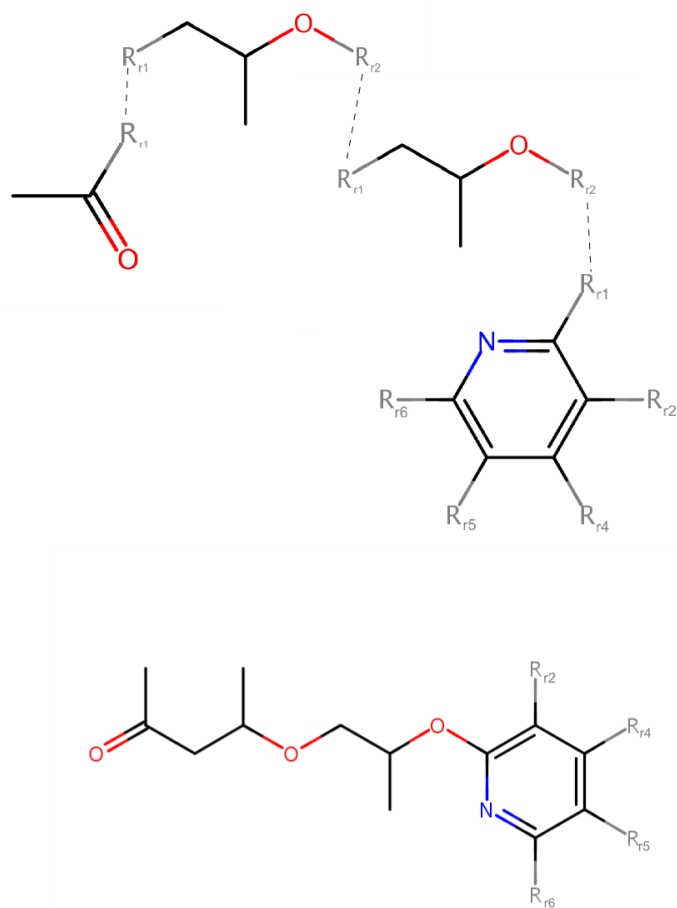


Figure 3.4: 2D representation of the process of joining the molecule in Example 1 and the completed molecule.

3.4 Example 2

In this example the use of `<fragmentList>` elements to contain `<fragment>` elements which are defined within the document for use elsewhere in the document is shown. The use of `<torsion>` elements to control torsion angles from bonds is also demonstrated.

Syntax used in this example:

- `<fragmentList>` element
- `<torsion>` element

3.4.1 `<fragmentList>`

A `<fragmentList>` element is used to contain `<fragment>` elements. A PML file can specify complex `<fragment>` elements in an initial `<fragmentList>` which are then referenced elsewhere in the PML file. These `<fragment>` elements can involve recursion. In this example (Table 3.4) a `<fragmentList>` is used to define some larger fragments that are then referenced later. The first `<fragment>` after the `<fragmentList>` is the start of the actual molecule. The `<fragment>` on line 10 contains two `<fragment>` and one `<join>` children:

```
10      <fragment id="eocl">
11        <fragment>
12          <molecule ref="g:eo" />
13        </fragment>
14        <join moleculeRefs2="PREVIOUS NEXT" atomRefs2="r1 r1" />
15        <fragment>
16          <molecule ref="g:c1" />
17        </fragment>
18      </fragment>
```

This means that when a later `<fragment>` references `eocl`, i.e. `<fragment ref="eocl">`, a copy of the combined fragments is made. (Figure 3.5). This combined fragment has an attachment point labelled `"r2"`. On line 38 in Example 2 when a `<fragment ref="eocl">` is used it is joined using the `<join moleculeRefs2="PARENT CHILD" atomRefs2="r4 r2">` element on line 35. The `'r2'` used in this join refers to the remaining `'r2'` in the combined

```

1 <fragment xmlns="http://www.xml-cml.org/schema"
2   xmlns:g="http://www.xml-cml.org/mols/geom1">
3   <fragmentList>
4     <fragment id="eo">
5       <molecule ref="g:eo" />
6     </fragment>
7     <fragment id="po">
8       <molecule ref="g:po" />
9     </fragment>
10    <fragment id="eocl">
11      <fragment>
12        <molecule ref="g:eo" />
13      </fragment>
14      <join moleculeRefs2="PREVIOUS NEXT" atomRefs2="r1 r1" />
15      <fragment>
16        <molecule ref="g:cl" />
17      </fragment>
18    </fragment>
19    <fragment id="eopoeo">
20      <fragment ref="eo" />
21      <join atomRefs2="r1 r2" moleculeRefs2="PREVIOUS NEXT" />
22      <fragment ref="po" />
23      <join atomRefs2="r1 r2" moleculeRefs2="PREVIOUS NEXT" />
24      <fragment ref="eo" />
25    </fragment>
26  </fragmentList>
27  <fragment>
28    <molecule ref="g:triazine">
29      <join moleculeRefs2="PARENT CHILD" atomRefs2="r2 r2">
30        <torsion>45</torsion>
31        <fragment>
32          <fragment ref="po" />
33        </fragment>
34      </join>
35      <join moleculeRefs2="PARENT CHILD" atomRefs2="r4 r2">
36        <torsion>45</torsion>
37        <fragment>
38          <fragment ref="eocl" />
39        </fragment>
40      </join>
41      <join moleculeRefs2="PARENT CHILD" atomRefs2="r6 r2">
42        <torsion>45</torsion>
43        <fragment>
44          <fragment ref="eopoeo" />
45        </fragment>
46      </join>
47    </molecule>
48  </fragment>
49 </fragment>

```

Table 3.4: Example 2 PML listing.

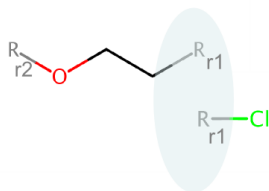


Figure 3.5: Representation of the **eoc1** fragment from line 10 in example 2 (Table 3.4)

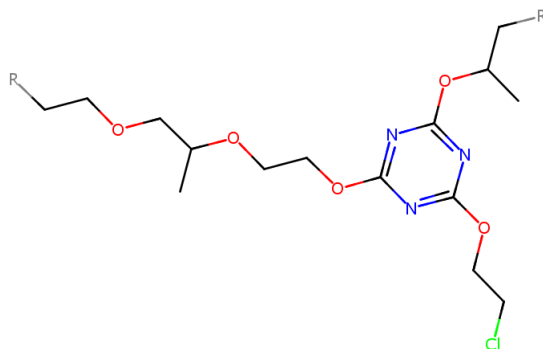


Figure 3.6: Representation of complete molecule from Example 2 (Table 3.4)

eoc1 fragment. Since the **eoc1** fragment is joined as a ‘CHILD’ the **eoc1** fragment is added as a side-chain to the ‘PARENT’ triazine fragment.

The completed molecule can be seen in Figure 3.6.

3.4.2 <torsion> elements

In Example 2 (Table 3.4) on line 30 a <torsion> element is used. These elements are used to specify the torsion angle when two <molecule> elements are joined together with a <join> element. The <torsion> element should be an XML child of the <join> element it refers to. The XML text value of the <torsion> element should be the angle of the torsion in degrees.

```
29      <join moleculeRefs2="PARENT CHILD" atomRefs2="r2 r2">
30          <torsion>45</torsion>
```

```

31         <fragment>
32             <fragment ref="po" />
33         </fragment>
34     </join>

```

Here the torsion angle has been set to 45°. Note that in the case of a `<join>` element with `moleculeRefs2="PARENT CHILD"` attribute both the `<torsion>` element and the CHILD `<fragment>` element are XML children of the `<join>` element. In the case of a `<join>` element with `moleculeRefs2="PREVIOUS NEXT"` only the `<torsion>` element would be a child of the join:

```

1  <join atomRefs2="r1 r2" moleculeRefs2="PREVIOUS NEXT">
2      <torsion>angle</torsion>
3  </join>

```

where the *angle* is the desired torsion angle in degrees.

In order to define a torsion angle it is necessary to specify four atoms. At the moment only two atoms have been specified. The way this is accomplished in PML is for the molecule files defining a CML molecule to include a `<label>` element with a `dictRef="cml:torsionEnd"` attribute. This element is added as an XML child of a desired atom in the molecule while its value specifies which R-group the torsion end is for:

```

1  <atom id="a3" elementType="C" x3="-0.703000" y3="-0.593000" z3="0.302000">
2      <label dictRef="cml:torsionEnd">r1</label>
3  </atom>

```

Here the atom with `id="a3"` is the torsion end for the `r1` group. If both the R-groups being joined together have a `cml:torsionEnd` specified then this defines the set of four atoms for the torsion angle. The torsion angle is then taken between the line from the `torsionEnd` atom and the atom attached to the R group to be joined, to the line between the other atom attached other R group to be joined and the second `torsionEnd` atom as show in Figure 3.7.

3.5 markushMixture

The previous two examples were completely deterministic. PML also allows for a statistical description of a molecule. One way of obtaining this

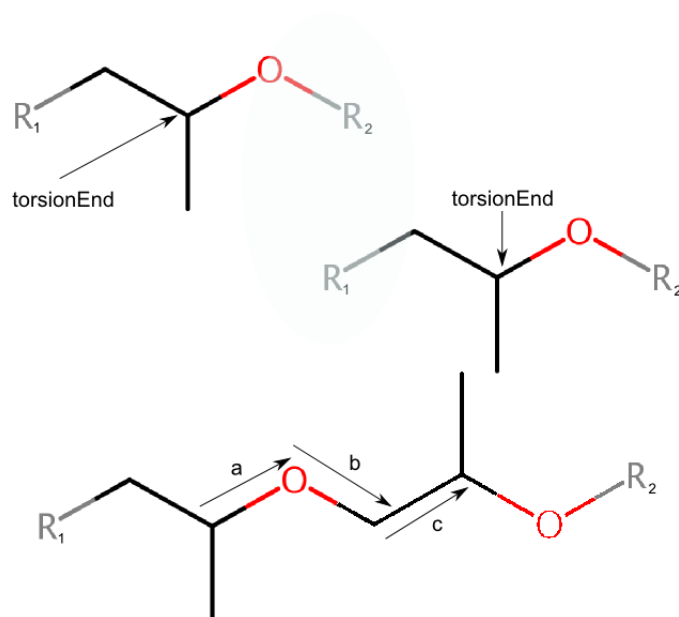


Figure 3.7: How a torsion angle can be defined between two fragments that are joined together. The torsion angle is the angle between the vectors *a* and *c* in the diagram.

is through the use of a `<fragmentList>` with a `role="markushMixture"` attribute. The syntax for this is as follows:

```

1      <fragmentList role="markushMixture">
2          <fragment ref="molA">
3              <scalar dictRef="cml:ratio" dataType="xsd:double">0.2</scalar>
4          </fragment>
5          <fragment ref="molB">
6              <scalar dictRef="cml:ratio" dataType="xsd:double">0.8</scalar>
7          </fragment>
8      </fragmentList>

```

Here the meaning of the `<fragmentList>` is modified by the `role="markushMixture"` attribute. Now the `<fragmentList>` will expand to one of its `<fragment>` children chosen at random. The relative probabilities of the different fragments are given by a `<scalar>` element with a double. In this example the relative probabilities add up to 1, but sum can be any number as the probabilities are relative rather than absolute. The ratios in this example mean that 80% of the time this `<fragment>` will represent a `<fragment ref="molB">` and 20% of the time a `<fragment ref="molA">`.

Any number of `<fragment>` elements can be used as the XML children of a `<fragmentList role="markushMixture">`, one of the `<fragment>` elements will be chosen at random based on the relative probabilities given. If a fixed result is desired a random number seed can be set in the JUMBO `fragmentTool` class in order to ensure repeatable results. This construct allows for complex behaviour. In a statistical co-polymer the `markushMixture` construct is used heavily (Section 3.6.2).

3.6 PolymerBuilder Templates

The polymer builder is a web application which uses PML to create 3D representations of a macromolecule. This was created in the course of this work in conjunction with David Jessop and Jennifer Ryder and forms the Polymer Builder component of Polymer Informatics Knowledge System (PIKS) in Figure 1. It uses HTML pages to interface with the user. The simplest example is the fixed length homopolymer template. The HTML page for this is shown in Figure 3.8. The PolymerBuilder can also generate statistical co-polymers which are discussed below (Section 3.6.2).

The HTML form allows the user to specify the repeat unit, the degree of polymerisation, the torsion angle, the starting group and the termination group. The values given by the user are used to set key-value pairs for an Extensible Stylesheet Language Transformations (XSLT) processor[37]. This XSLT processor reads in an XSL stylesheet (in this case for a homopolymer) and the values from the HTML form are parsed in as XSL parameters. The XSLT processor then produces the PML output. The PML is then processed with JUMBO into a CML document. This process is shown in Figure 3.9.

3.6.1 Fixed length homopolymer

A fixed length homopolymer consists of a fixed number of one type of repeat unit. The full XSLT for the fixed length homopolymer template is shown below. For subsequent examples the XSLT will be omitted and only the PML template will be shown.

Fixed length linear homopolymer

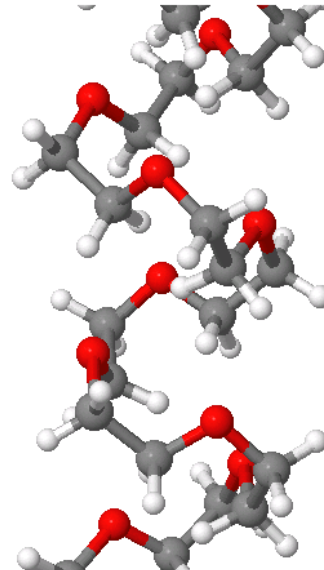
R1  R2

Degree of polymerisation

Torsion angle

End group 1	Repeat Unit	End group 2
<input type="radio"/> Acetoxy <input type="radio"/> Acetyl <input type="radio"/> Bromine <input type="radio"/> Chlorine <input type="radio"/> Ethyl <input type="radio"/> Fluorine <input type="radio"/> Hydrogen <input type="radio"/> Hydroxy <input type="radio"/> Methoxycarbonyl <input type="radio"/> Methyl <input type="radio"/> Propanoic Acid	<input checked="" type="radio"/> Acrylic Acid <input type="radio"/> cis-Acetylene <input type="radio"/> trans-Acetylene <input type="radio"/> Acrylonitrile <input type="radio"/> para-Benzene <input type="radio"/> Difluoroethylene <input type="radio"/> Dimethylbutene <input type="radio"/> Dimethylsiloxane <input type="radio"/> Ethylene <input type="radio"/> Ethylene Oxide <input type="radio"/> Ethylene Glycol <input type="radio"/> α-D-glucose <input type="radio"/> Glycine <input type="radio"/> L-Alanine <input type="radio"/> Methyl Methacrylate <input type="radio"/> Naphthalene <input type="radio"/> Propylene Oxide <input type="radio"/> Propylene <input type="radio"/> Propylene Glycol <input type="radio"/> Styrene <input type="radio"/> Triazene <input type="radio"/> Vinylalcohol <input type="radio"/> Vinylchloride	<input type="radio"/> Acetoxy <input type="radio"/> Acetyl <input type="radio"/> Bromine <input type="radio"/> Chlorine <input type="radio"/> Ethyl <input type="radio"/> Fluorine <input type="radio"/> Hydrogen <input type="radio"/> Hydroxy <input type="radio"/> Methoxycarbonyl <input type="radio"/> Methyl <input type="radio"/> Propanoic Acid

Unilever
Cambridge
Centre For Molecular Science Informatics



Powered by JUMBO



Figure 3.8: Screenshot of the HTML interface for the polymer builder web-service. This is the template for a fixed length homopolymer.

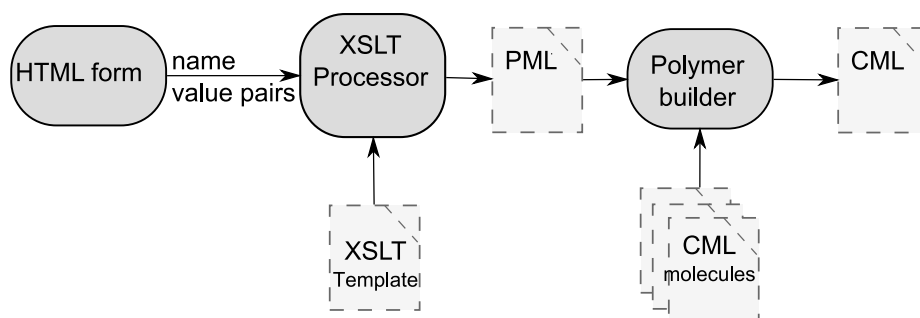


Figure 3.9: The process of building a CML document from the HTML form.

```

1 <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
2   <xsl:param name="count">10</xsl:param>
3   <xsl:param name="molecule">g:eo</xsl:param>
4   <xsl:param name="end1">g:h</xsl:param>
5   <xsl:param name="end2">g:h</xsl:param>
6   <xsl:param name="torsion">180</xsl:param>

```

The initial lines 2-6 are used only to set default values for the parameters. This means that if the user did not specify an option from the HTML the XSLT transform will supply default values rather than failing to complete.

```

7   <xsl:template match="/">

```

Line 7 is used to insert the desired XML into a new document, a dummy empty document is used which then is replaced with the contents of the `<xsl:template>` element when the XSLT transform. Since any document will match `"/` the contents of the dummy document is irrelevant.

```

8       <fragment convention="cml:PML-basic"
9         xmlns="http://www.xml-cml.org/schema"
10        xmlns:g="http://www.xml-cml.org/mols/geom1">
11         <fragment>
12           <molecule ref="{ $end1}" />
13         </fragment>
14         <join atomRefs2="r1 r1" moleculeRefs2="PREVIOUS NEXT" />
15         <fragment countExpression="*({ $count})">
16           <join atomRefs2="r2 r1" moleculeRefs2="PREVIOUS NEXT">
17             <torsion><xsl:value-of select="$torsion"/></torsion>
18           </join>
19           <fragment>
20             <molecule ref="{ $molecule}" />
21           </fragment>
22         </fragment>
23         <join atomRefs2="r2 r1" moleculeRefs2="PREVIOUS NEXT" />
24         <fragment>

```

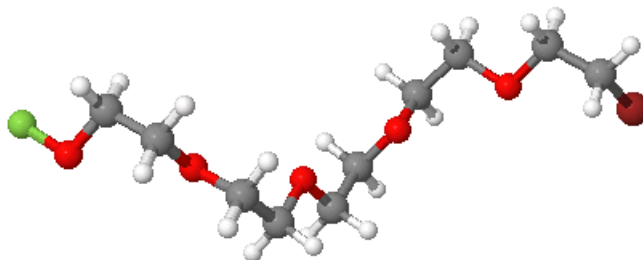


Figure 3.10: A 3D representation of an example molecule created using the fixed length homopolymer template. The settings used were bromine for the starting group, fluorine for the terminating group, ethylene oxide for the repeat unit, degree of polymerisation 5 and a torsion angle of 180 degrees.

```

25         <molecule ref="{ $end2}" />
26     </fragment>
27 </fragment>

```

Lines 8-27 show the PML document that will be produced. However, this document is not yet complete as the XSLT transform will replace the variables written in the form “`{ $name }`” with the value of the variable *name*. For example, if the starting end group had been set to fluorine, then the XSL parameter `end1` would have the value `g:f`. This would cause the `{ $end1 }` to be replaced by `g:f` when the XSTL processor ran.

On line 17 a different syntax is used to insert a value from an XSL parameter. This different construct is used as we are now inserting an element value rather than an attribute value.

```

28 </xsl:template>
29 </xsl:stylesheet>

```


The final lines close the stylesheet and template.

An example of a 3D rendering of the PML output from this template is shown in Figure 3.10.

3.6.2 Statistical copolymer

Statistical copolymers consist of at least two different repeat units where there is a known statistical relationship describing the ordering of repeat units

Variable Length Statistical Copolymer



Probability A after A

Probability B after A

Probability of termination after A

Probability A after B

Probability B after B

Probability of termination after B

Torsion angle

Figure 3.11: Screenshot of the HTML interface for the statistical copolymer template.

along the macromolecule depending on the previous repeat unit. Statistical copolymers can be used to create artificial membranes, for example for use in fuel cells [38]. The HTML interface for the statistical copolymer is shown in Figure 3.11. This template allows the creation of a random copolymer. The relative reactivity ratios of the two monomers can be approximated as probabilities of the repeat units given the previous repeat unit. This approach requires the use of recursion and `markushMixture` attributes.

A user can specify the starting group, ending group, two different molecular fragments (referred to as molA and molB) and the probabilities of a molA following a molA, a molA following a molB, a molB following a molA and a molB following a molB. A termination probability can also be set to produce a random length of chain. Since this only depends on the current state it is formally a Markov chain, taking the last added fragment as the current state.

The variables which the user has entered into the HTML form are transferred as variables into the XSLT stylesheet in the same process as for the fixed length homopolymer (Section 3.6.1). These variables are then used to

replace the text in the template with the information the user has entered. For example, if the radio button for “Repeat Unit A” was set to “Acrylic Acid” then `g:acryl` would replace all instances of `{ $\$$ moleculeA}` in the template.

The initial XSLT elements are not shown for simplicity. Since this template is quite long, it has been broken into smaller sections in the discussion below to aid understanding.

The start of the resulting PML document is a `<fragmentList>` that defines a range of different fragments. These `<fragment>` elements have the values from the HTML form inserted into the `ref` attributes for their `<molecule>` child:

```

1      <fragment convention="cml:PML-basic"
2      xmlns="http://www.xml-cml.org/schema"
3      xmlns:g="http://www.xml-cml.org/mols/geom1">
4      <fragmentList>
5          <fragment id="end1">
6              <molecule ref="{ $\$$ end1}" />
7          </fragment>
8          <fragment id="end2">
9              <molecule ref="{ $\$$ end2}" />
10         </fragment>
11         <fragment id="molA">
12             <molecule ref="{ $\$$ moleculeA}" />
13         </fragment>
14         <fragment id="molB">
15             <molecule ref="{ $\$$ moleculeB}" />
16         </fragment>

```

Here, four different fragments have been defined: `end1`, `end2`, `molA` and `molB`. These `<fragment>` elements are then referred to later:

```

17         <fragment id="AB">
18             <fragment ref="molB" />
19             <join atomRefs2="r2 r1" moleculeRefs2="PREVIOUS NEXT">
20                 <torsion><xsl:value-of select="{ $\$$ torsion}" /></torsion>
21             </join>
22             <fragment ref="BX" />
23         </fragment>

```

In these lines of PML a new `<fragment>` with `id="AB"` has been defined. This `<fragment>` consists of the previously defined `molB` `<fragment>` followed by a `<join>` element linking it to an as yet undefined `id="BX"` `<fragment>` element which is defined on line 64 below. Further `<fragment>` elements are then defined:

```

24     <fragment id="AA">
25         <fragment ref="molA" />
26         <join atomRefs2="r2 r1" moleculeRefs2="PREVIOUS NEXT">
27             <torsion><xsl:value-of select="$torsion"/></torsion>
28         </join>
29         <fragment ref="AX" />
30     </fragment>
31     <fragment id="BA">
32         <fragment ref="molA" />
33         <join atomRefs2="r2 r1" moleculeRefs2="PREVIOUS NEXT">
34             <torsion><xsl:value-of select="$torsion"/></torsion>
35         </join>
36         <fragment ref="AX" />
37     </fragment>
38     <fragment id="BB">
39         <fragment ref="molB" />
40         <join atomRefs2="r2 r1" moleculeRefs2="PREVIOUS NEXT">
41             <torsion><xsl:value-of select="$torsion"/></torsion>
42         </join>
43         <fragment ref="BX" />
44     </fragment>

```

Here we have defined the AA, BA and BB fragments. These `<fragment>` elements are very similar to the AB `<fragment>` defined above. They all consist of either a molA or molB `<fragment>` element joined with a `<join>` element to either a BX or AX `<fragment>` element.

```

45     <fragment id="AX">
46         <fragmentList role="markushMixture">
47             <fragment ref="molA">
48                 <scalar dictRef="cml:ratio" dataType="xsd:double">
49                     <xsl:value-of select="$pAT"/>
50                 </scalar>
51             </fragment>
52             <fragment ref="AA">
53                 <scalar dictRef="cml:ratio" dataType="xsd:double">
54                     <xsl:value-of select="$pAA"/>
55                 </scalar>
56             </fragment>
57             <fragment ref="AB">
58                 <scalar dictRef="cml:ratio" dataType="xsd:double">
59                     <xsl:value-of select="$pAB"/>
60                 </scalar>
61             </fragment>
62         </fragmentList>
63     </fragment>

```

Here we have defined the first of the two remaining `<fragment>` elements.

This AX fragment contains a `<fragmentList>` element with a `role="markushMixture"`

attribute. When this stochastic element is built it will therefore either become a `<fragment ref="molA">`, a `<fragment ref="AB">` or a `<fragment ref="AA">`. If a `<fragment ref="molA">` is chosen no recursion can be done; the molecule has stopped growing. If a `<fragment ref="AA">` is chosen then the initial `AX` fragment is replaced by an `AA` fragment. But as we have previously seen, this `AA` fragment is defined to be a `molA` fragment joined to an `AX` fragment. This means that our `AX` fragment has been replaced by a `molA + AX`. The new `AX` fragment can then be expanded in the same way leading to a recursive growth of the document until termination occurs. This process is shown diagrammatically in Figure 3.12.

The probabilities of termination, addition of repeat unit A and addition of repeat unit B are controlled by the parameters set by the user in the HTML form, which replace the strings `$pAT`, `$pAA` and `$pAB`.

```

64     <fragment id="BX">
65         <fragmentList role="markushMixture">
66             <fragment ref="molB">
67                 <scalar dictRef="cml:ratio" dataType="xsd:double">
68                     <xsl:value-of select="$pBT"/>
69                 </scalar>
70             </fragment>
71             <fragment ref="BA">
72                 <scalar dictRef="cml:ratio" dataType="xsd:double">
73                     <xsl:value-of select="$pBA"/>
74                 </scalar>
75             </fragment>
76             <fragment ref="BB">
77                 <scalar dictRef="cml:ratio" dataType="xsd:double">
78                     <xsl:value-of select="$pBB"/>
79                 </scalar>
80             </fragment>
81         </fragmentList>
82     </fragment>
83 </fragmentList>

```

The `BX <fragment>` element has been defined in an analogous way to the definition of the `AX <fragment>` element.

At this stage all the `<fragment>` elements have been defined and the initial `<fragmentList>` has been closed. Next the PML document initialises the recursive molecule:

```

84     <fragment>
85         <fragment ref="end1"/>

```

```

86         <join atomRefs2="r1 r1" moleculeRefs2="PREVIOUS NEXT" />
87         <fragment ref="AX" />
88         <join atomRefs2="r2 r1" moleculeRefs2="PREVIOUS NEXT" />
89         <fragment ref="end2"/>
90     </fragment>
91 </fragment>

```

In this section the molecule starts with a `<fragment ref="end1">` element, which is the starting end group defined on line 5. This is connected to an `AX <fragment>` with a `<join>` element. The `AX <fragment>` is connected to the final terminating group, an `end2 <fragment>` that was defined on line 7. The `AX <fragment>` is expanded recursively until termination occurs, (when it is finally joined to the terminating end group defined on line 8) following its definition on line 45. An example of the output of this template is shown in Figure 3.13.

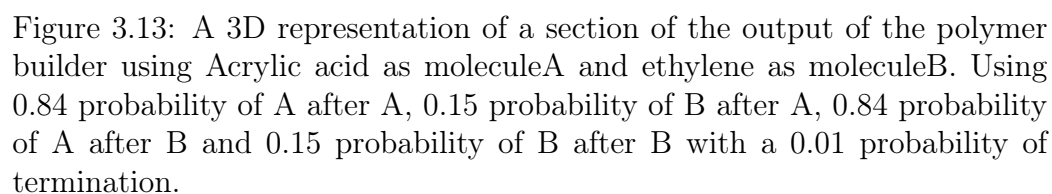
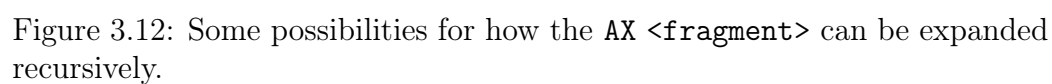
If no termination probability is set then this would never terminate. When the `FragmentTool` class from JUMBO is used to convert such a PML document into CML, a ceiling can be set that will force termination of a recursive `<fragment>` element when a set number of iterations are reached.

3.7 JUMBO

The JUMBO[36] library includes tools to process PML files. These tools can be called easily from any Java program. The class `FragmentTool` is the main entry point to the PML code. Example code to build a CML molecule from a PML file is shown in Table 3.5. In this code a catalog is being used to resolve QNames of `<molecule>` elements to a directory. These `<molecule>` elements could be explicitly stated inside the PML document to avoid this.

Here the `CMLBuilder` is used to load the input file, and the `FragmentTool` is the class that processes the PML document into a CML molecule. The `ResourceManager` is used to read the catalog and transfer the information to the `FragmentTool`.

In order to use the JUMBO Java library in Java projects it is necessary to add the JAR file to the build path. A useful method for managing Java build paths is through the use of Apache Maven and POM files. The POM



```

1 CMLFragment fragment=null;
2 CMLBuilder builder = new CMLBuilder();
3 Document doc=null;
4 try {
5     doc=builder.build("in.xml");
6 } catch (Exception e) {
7     e.printStackTrace();
8 }
9 fragment=(CMLFragment)doc.getRootElement();
10 FragmentTool tool = FragmentTool.getOrCreateTool(fragment);
11 File CMLMapFile = new File("src/main/resources/catalog.xml");
12 ResourceManager resourceManager = new ResourceManager(CMLMapFile.toURI());
13 tool.processAll(resourceManager);
14 fragment.detach();
15 Document out = new Document(fragment);
16 Serializer ser = new Serializer(new FileOutputStream("out.cml"));
17 ser.write(out);
18 ser.flush();

```

Table 3.5: Java code to use JUMBO `FragmentTool` to build a CML document from a PML document.

file to include JUMBO is given below:

```

1     <dependency>
2         <groupId>cml</groupId>
3         <artifactId>jumbo</artifactId>
4         <version>5.5.1-SNAPSHOT</version>
5     </dependency>

```

this instructs Maven to import JUMBO and the dependencies JUMBO requires, and include them in the classpath.

3.8 Extension to PML to allow for depletion of monomer

While the existing PML language allows for a huge degree of expressibility to define the macromolecules a PML document represents, the language could only represent a Markov-process. In order to allow the creation of ensembles of macromolecules which have differing probabilities along the chain, new syntax and processing tools had to be developed.

When a statistical co-polymer is synthesised there may be a depletion of reagents throughout the reaction. PML assumes that all reagents are present in infinite concentrations without depletion. If one reactant is much more reactive than the other, rapid depletion of the more reactive monomer can lead to a block copolymer[39]. In order to represent this in PML some additions to the language were required. The use of an element called `<amount>` was added to address this limitation. In this work, an `<amount>` element can be added as a child of a `<fragment>`. Then, later in the document when a `<fragmentList role="markushMixture">` element references the `<fragment>` with a `rateDepend` attribute the relative amounts of the various fragments are factored into the statistical probabilities, and the amount of the chosen fragment is decremented.

Because the `<amount>` elements are stored on the `<fragment>` elements referenced in the markush mixtures, multiple markush mixture lists can reference the same `<fragment>` with the same `<amount>` attached. This allows for a statistical co-polymer where the probability of adding a new `<fragment>` depends on both the previously added `<fragment>` and the relative amounts of the competing `<fragment>` elements. If one of the `<fragment>` elements does not have an `<amount>` element, then the probability is assumed to be fixed and will not vary as the other reactants are used up. This can be used to set a fixed termination probability.

The equation used to determine a new probability is shown below:

$$P'_i = \frac{P_i x_i}{x_t} \times \frac{\sum_{i=1}^n P_i}{\sum_{i=1}^n \frac{P_i x_i}{x_t}}$$

where $x_t = \sum_{i=1}^n x_i$, P_i is the old probability of the i th fragment, x_i is the amount of the i th fragment and P'_i is the new probability of the i th fragment. If a fragment has no amount, then $P'_i = P_i$. This is distinct from the case where $x_i = 0$ in which case the formula above gives $P'_i = 0$. In the event that $x_t = 0$ then x_i/x_t will be undefined. In this case $P'_i = 0$ for all fragments with $x_i = 0$, and $P'_i = P_i$ for the fragments with no amount. In the usual case where the only fragment without an amount is a termination step, this will force a termination when the fragments representing monomers have been

fully depleted.

3.8.1 Example usage

An example PML file for a poly[ethene-co-(prop-1-ene)] copolymer is given below:

```
1 <fragment xmlns="http://www.xml-cml.org/schema" xmlns:g="http://www.xml-cml.org/mols/geom1">
2   <fragmentList>
3     <fragment id="ethyl">
4       <amount>750</amount>
5       <molecule ref="g:ethylene">
6         </molecule>
7     </fragment>
```

Here the `<amount>` element has been used to specify the number of available repeat units of ethylene. Whenever an *ethyl* `<fragment>` is dereferenced in the document, this number will be decremented.

```
8     <fragment id="propyl">
9       <amount>300</amount>
10      <molecule ref="g:propylene" />
11    </fragment>
```

Similarly, here the `<amount>` element has been used to specify the number of available repeat units of propylene. Whenever a *propyl* `<fragment>` is dereferenced in the document, this number will be decremented.

```
12    <fragment id="PX">
13      <fragment ref="propyl" />
14      <join atomRefs2="r1 r2" moleculeRefs2="PREVIOUS NEXT" />
15      <fragment ref="propylX" />
16    </fragment>
17    <fragment id="EX">
18      <fragment ref="ethyl" />
19      <join atomRefs2="r1 r2" moleculeRefs2="PREVIOUS NEXT" />
20      <fragment ref="ethylX" />
21    </fragment>
```

Here the PX and EX `<fragment>` elements are fulfilling an analogous role to the AA, BB elements from Section 3.6.2. They add either a *propyl* or *ethyl* `<fragment>`

```
22    <fragment id="ethylX">
23      <fragmentList role="markushMixture">
24        <fragment ref="ethyl">
25          <scalar dictRef="cml:ratio" dataType="xsd:double">0.01</scalar>
```



```

26         </fragment>
27         <fragment ref="EX" rateDepend="ethyl">
28             <scalar dictRef="cml:ratio" dataType="xsd:double">0.77</scalar>
29         </fragment>
30         <fragment ref="PX" rateDepend="propyl">
31             <scalar dictRef="CML:rato" dataType="xsd:double">0.22</scalar>
32         </fragment>
33     </fragmentList>
34 </fragment>
35 <fragment id="propylX">
36     <fragmentList role="markushMixture">
37         <fragment ref="propyl">
38             <scalar dictRef="cml:ratio" dataType="xsd:double">0.01</scalar>
39         </fragment>
40         <fragment ref="EX" rateDepend="ethyl">
41             <scalar dictRef="cml:ratio" dataType="xsd:double">0.77</scalar>
42         </fragment>
43         <fragment ref="PX" rateDepend="propyl">
44             <scalar dictRef="CML:rato" dataType="xsd:double">0.22</scalar>
45         </fragment>
46     </fragmentList>
47 </fragment>

```

Here the *propylX* and *ethylX* fragments have been defined in a similar fashion to the *AX* and *BX* fragments in Section 3.6.2.

```

48     <fragment id="start">
49         <fragmentList role="markushMixture">
50             <fragment ref="propylX">
51                 <scalar dictRef="cml:ratio" dataType="xsd:double">0.77</scalar>
52             </fragment>
53             <fragment ref="ethylX">
54                 <scalar dictRef="cml:ratio" dataType="xsd:double">0.22</scalar>
55             </fragment>
56         </fragmentList>
57     </fragment>
58 </fragmentList>

```

In this section the *start* **<fragment>** element is defined. This expands to either a *propylX* or a *ethylX* randomly. This is to ensure that the probabilities are not being biased by starting all the chains with the same fragment.

```

59     <fragment id="f0">
60         <fragment ref="start" />
61     </fragment>
62 </fragment>

```

Finally the molecule is started with a *start* **<fragment>** element. This will then be recursively expanded until the molecule terminates. The difference

between this and the statistical copolymer PML document in Section 3.6.2 is that the probabilities here change as the macromolecule grows.

3.8.2 Comparison with an Experimental Sample

The extension to PML described in Section 3.8 allowed the creation of a PML document which produces ensembles of macromolecules which fit the distributions produced by experiments. With modern characterisation techniques the distribution of co-monomers along a polymer chain can be measured. Suárez et al.[40] manufacture a range of ethylene/propylene copolymers with differing compositions. These are then characterised using GPC-IR to find how the composition of the macromolecules varies with molecular weight, M_w .

3.8.3 Methods

To see if the extended PML could represent the experimental data produced by Suárez et al. [40] the extension to PML was implemented in JUMBO. The PML document shown in Section 3.8.1 was processed 5,000 times to produce an ensemble of macromolecules. This results in 5,000 CML files, each representing a macromolecule explicitly. This collection of CML files is quite large, taking up 953 MB (167 MB zipped).

An analysis tool written in Java was then used to parse each file and determine the molecular weight of the sample, and the ratio of propylene to ethylene derived repeat units in that macromolecule. This resulted in a file with 5,000 datapoints for molecular weight and propylene to ethylene ratio.

In order to compare this to the experimental data plots, it was necessary to perform some computational transformations on the data to produce a similar plot. To produce the GPC-like plot the data points were binned into equal-width bins by $\log(\text{molecular weight})$. The total mass of macromolecules in each bin was then calculated to give an absorbance for that bin. The proportion of ethylene was calculated as the average proportion of ethylene derived repeat units in the macromolecules in the bin. This data was then

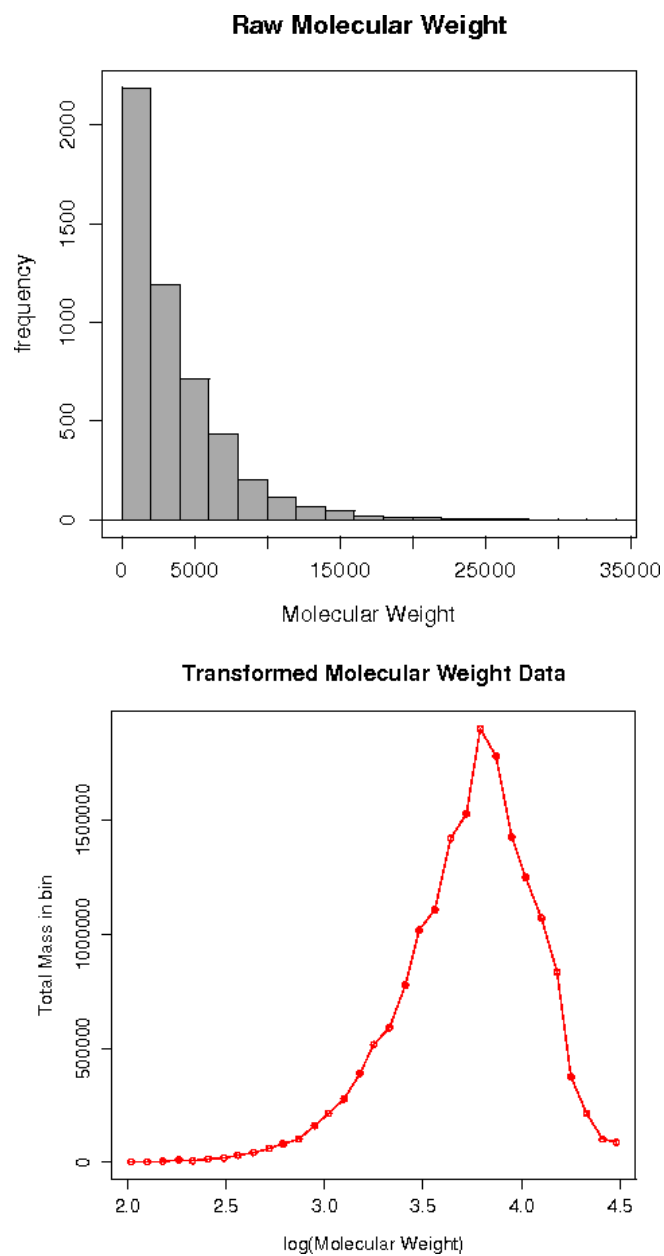


Figure 3.14: The raw molecular weight data compared with the GPC-like plot produced after transformation.

plotted to give a comparable graph to the experiment. In Figure 3.14 the differences between the raw and processed molecular weight is shown.

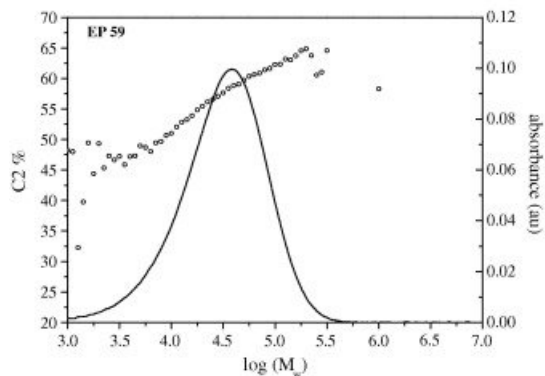
3.8.4 Results

An example of the experimental data produced by GPC-IR[40] is shown in Figure 3.15(a). In this sample of polymer, the ratio of ethylene to propylene derived repeat units depends on the molecular weight of the macromolecules. This type of ensemble of macromolecules can now be represented using the extension to PML.

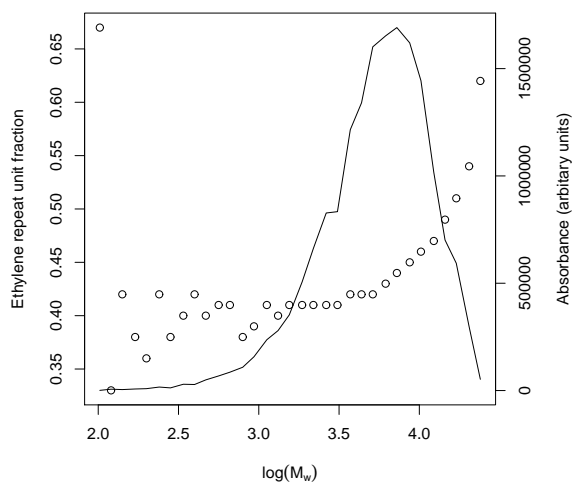
In Figure 3.15(b) the results of the PML are shown. The M_w of the experimental sample was 61,967 while the M_w for the PML document produced sample was 6,755 which resulted in an x-axis difference of 0.96 log units. The molecular weight could be increased by lowering the termination probability by a factor of ten, but would then take approximately ten times as long to run and produce outfile files ten times as large.

3.9 Conclusions

This chapter has described Polymer Markup Language, and introduced an extension to PML which allows the representation of a wide-range of copolymers, where the composition of the macromolecules depends on the molecular weight in some way. In Figure 3.15(a) a well characterised experimental sample where the composition does depend on molecular weight is shown. In Figure 3.15(b) the output of the PML document given in Section 3.8 is shown. While the exact distribution of the ethylene percentage is not the same between the two, the PML document has a very similar molecular weight distribution and the same increasing trend of ethylene percentage increasing with increasing molecular weight. This shows that the extension to PML can represent polymer samples where the distribution of repeat units varies with the molecular weight. Both plots show a value over 60% at the high end of the molecular weight distribution and around 40% at the low end. The experimental data has a more linear increase than the simulated



(a) A Figure from [40] showing how the Ethylene repeat unit content changes with molecular weight for an ethylene/propylene copolymer produced with metallocene catalysts. The dots show the % ethylene in the macromolecules while the line shows the absorbance of the GPC detector.



(b) A plot generated from 1000 macromolecules represented by the PML document in this section. The dots show the ethylene % and the line shows the total weight of macromolecules in a given weight bin.

Figure 3.15: Plots of experimental data in contrast to the output of a PML document. The M_w in Figure 3.15(a) is 61,967 while in Figure 3.15(b) it is 6,755 which results in an x-axis difference of 0.96 log units.

data.

The distribution of ethylene in the experimental sample is very likely to have factors other than the changing concentration of monomers over the course of the reaction such as different rates of diffusion of monomers, different mobilities of growing chains in solution and differing interactions with the catalyst.

As well as the ethylene-propylene copolymers described in Section 3.8.2 the extension can also be used to represent block co-polymers. If one of the monomers is much more reactive than the other, then the initial growth will be mostly the more reactive monomer, while the later stages of polymerisation will be mostly the less reactive monomer, producing a block copolymer [39]. This could be created from the PML document in Section 3.8 by changing the parameters.

In order to represent a sample of polymer, it is necessary to have sufficient characterisation data about that polymer sample. In Chapter 4 the PoLyInfo database as a source of polymer data is discussed.

Chapter 4

PoLyInfo

For an informatics system to be of use, a source of data is required. In this Chapter the data contained within the PoLyInfo database[41] is examined. This database was chosen as the data is free to access. There are other polymer databases, but they either charge for access[42, 43], or contain only very limited amounts of data[44, 45]. The aim was to see what information could be extracted from the publicly available resource, to what extent did the data in the database match the database’s documentation, and to compare how the aggregate data in the database compares to historical data correlations such as glass transition temperature being correlated to melting temperature [46].

The data in the PoLyInfo database was extracted using the spider component of PIKS (See Chapter 5) and added to a searchable XML database for querying. This was used to produce the analysis in this chapter. The data was also then used for validation (Chapter 6) and Modelling (Chapter 7).

The PoLyInfo database is run by the Japanese National Institute of Material Sciences (NIMS). The PoLyInfo database describes itself as follows[47]:

“Polymer Database PoLyInfo systematically provides various data required for polymeric material design. The main data source is academic literature on polymers. Information on polymers including properties, chemical structures, IUPAC names, processing methods of measured samples, mea-

surement conditions, monomers, and polymerization methods are stored in a object database. About 100 types of properties, including thermal, electrical and mechanical properties are covered. Homopolymers, copolymers, and polymer blend, composites and compounds that consist of homopolymers and copolymers are open to the public. Properties Estimation Subsystem and NMR spectral data are also available as advanced functions.”

4.1 History

PoLyInfo was set up by the Japan Science and Technology Agency (JST) in 1995. The system was first accessible to the public from January 1998 to March 2001 when the public beta was running. The PoLyInfo database was released in April 2001. Control was handed over to NIMS in 2003. The database has been enhanced over the years with additional features such as property prediction using group contribution methods. The data in the database are updated from time to time in large batches. Since June 2003 the database has been updated 13 times, with the most recent update being June 2009.

As can be seen in Figure 4.1, growth of the database seems to be linear, rather than the exponential growth often seen in other data repositories. This could either be due to a constant workforce entering the data from journals with an ever increasing backlog, or a linear rate of publishing on polymer data.

4.2 Content

Most data in PoLyinfo is aggregated from the literature. For data to be added the database is it must satisfy three conditions[41]:

1. Chemical structures of constitutional units (repeat units) are clearly determined.
2. Natural polymers such as proteins, polynucleotides and polysaccharides are not included in the database, except for polypeptides for which the

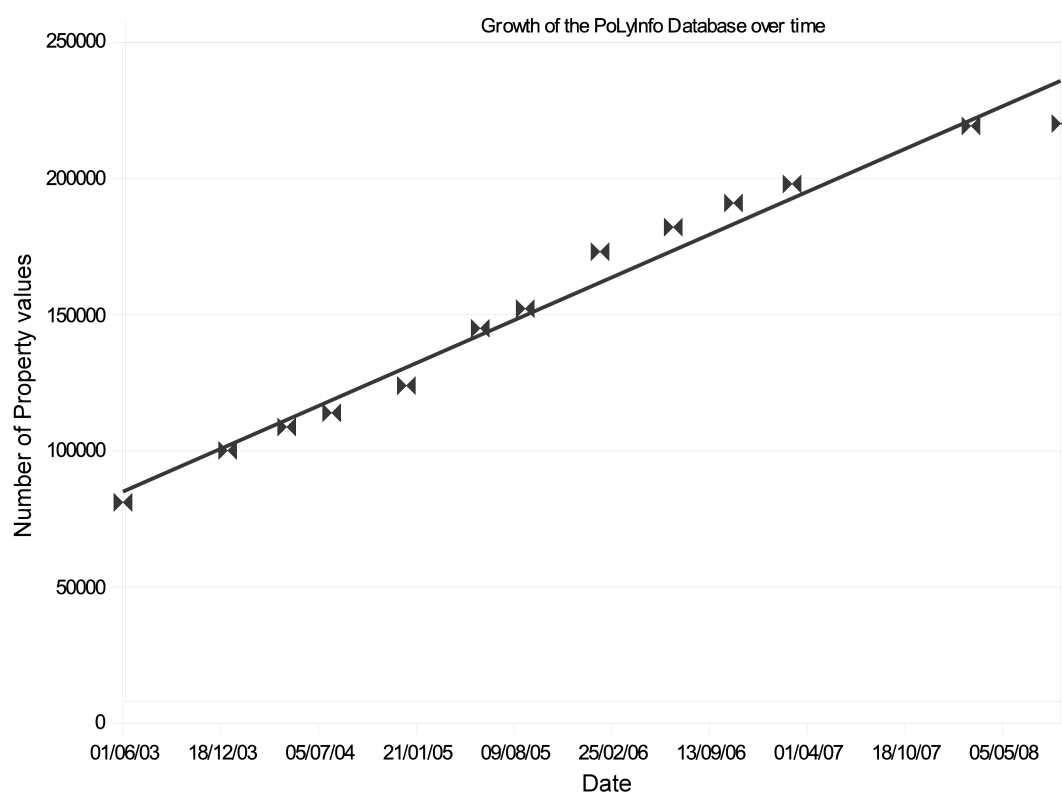


Figure 4.1: Graph to show how the number of sample property data points in the PoLyInfo database has changed over time. The data for this graph was taken from the historical data page on the website of the PoLyInfo database. A linear regression line shows approximately linear growth in the number of property points in the database.

constitutional units (CUs) are clearly determined.

3. Properties are measured by authors themselves and also both measurement methods and conditions are reported.

If data comply with these criteria, it is then added to the database. There is no mention of how automated this procedure is, but the process is not without errors; whether they are due to human or machine error cannot be easily determined. This is discussed in Chapter 6.

A diagram illustrating the data-model used is shown in Figure 4.2. This shows that the data feed in from the literature on the top-right hand side, where they are then added to the database. The database stores the information of that sample, including its preparation information and the polymer constitutional unit (repeat unit) which is stored as a list of subunits (used for the group contribution method property prediction.)

The IUPAC Nomenclature PoLyIndex component calculates the IUPAC source and structure-based names for the polymer. The monomers used to synthesise the polymer are also stored. The group contribution module calculates some polymer properties based on the group contribution values from the *sub-units* or fragments of the repeat unit. This means that the value predicted take no account of molecular weight or crystallinity of a sample.

4.3 Navigation

The PoLyInfo database is navigated via a web interface. The navigation sidebar (Figure 4.3) contains methods to browse and search the data. The *Property table* shows how many property data-points exist for each class of polymer. The properties are divided into the property classes of: *Thermal*, *Physical and physico-chemical*, *Electrical* and *Mechanical*. The polymer classes are shown in Table 4.1. These polymer classes can then be browsed by number of carbon atoms in the repeat unit.

The polymer classes are defined in terms of the repeat unit structure. A polymer that fits multiple class definitions is a member of both classes. The

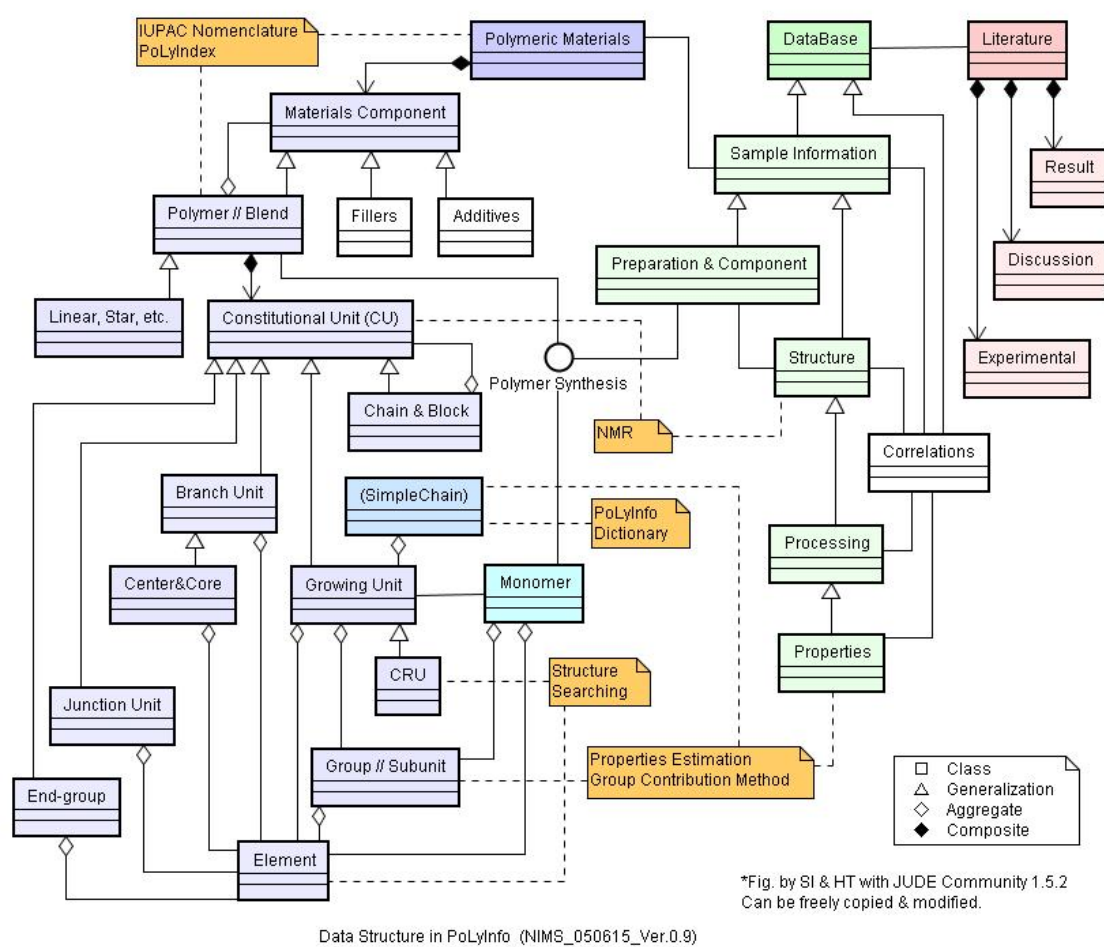


Figure 4.2: Diagram from the PoLyInfo website describing the data-structure used in the database.

Polymer classes	
polyolefins	polyureas
polystyrenes	polyimides
polyvinyls	polyanhydrides
polyacrylics	polycarbonates
polyhalo-olefins	polyimines
polydienes	polysiloxanes
polyoxides	polyphosphazenes
polysulfides	polyketones
polyesters	polysulfones
polyamides	polyphenylenes
polyurethanes	others

Table 4.1: Table of the different classes of polymers used in the PoLyInfo database.

English definitions on the PoLyInfo website are sometimes badly translated, but understandable. Some examples are shown below:

Polyolefins ‘Polymer composed only of saturated aliphatic hydrocarbon radical.’

Polystyrenes ‘Polymer that consists of the aliphatic hydrocarbon radical, and the aromatic ring is directly connected to at least one of the main chain carbon inside.’

Polyphenylenes ‘It consists of the element that a main chain is composed only of the hydrocarbon radical and Polymer on a main chain including the phenyl radical.’

Other polymers ‘All Polymer that doesn’t belong to classification 01-21’
(*This is number 22*)

22 Polymer Classes are defined, 21 of which are structural classes. The remaining class is a ‘catch all’ for polymers that do not fit into any other class. Polymers may belong to more than one structural class.

The *Popular polymer* link lists a selection of common polymers in the database. These polymers are selected by PoLyInfo by some method, possibly

PoLyInfo MENU

Help Japanese

Help English

Polymer

Easy browse

- [Property table](#)
- [Popular polymer](#)
- [Polymer blend](#)
- [Plotted data](#)

Search condition

- [Basic](#)
- [Advanced](#)

Structure search

- [By parts](#)
- [By substructure](#)

Property Prediction

- [GroupContribution](#)

Monomer

[Easy browse](#)

[Search](#)

Top

Figure 4.3: The navigation sidebar menu from the PoLyInfo database.

based on frequency of page views, or hand-selected to be common polymers. These polymers along with the number of property points for each one, can be seen in Table 4.2. This page makes the most popular polymers the fastest to access.

The *polymer blend* page lists different polymer blends which have been transcribed and the number of samples of each one. Also shown is the number of property data points recorded for blends of that type. For example there is one sample of a blend of poly(ethene) and poly(hex-1-ene). The characterisation data for the blend is rather sparse, however, and does not indicate what proportion of each polymer was used in the blend. For this reason polymer blends were not investigated further in this Thesis.

The *basic search* page (Figure 4.4) shows the various options for a basic search. The polymer or material name drop down box contains the names of some common polymers to aid searching. Either one of these can be selected or a name entered into the box provided. The material type facilitates search for samples of a specific material, either a *neat resin*, *composite*, *compound* or *composite and compound*. Polymer type lets one search for either homopolymers or copolymers. The polymer class selection box contains the polymer classes previously shown in Table 4.1. The “CU Formula of Homopolymer or a Component” part of the form lets you enter a molecular formula to search for in the CU of the polymer. The property box allows you to search for a property within a user-defined range. This allows you to search for samples of a polymer that have specific properties (Samples are discussed more in Section 4.7). The reference section lets you search for samples of polymers that were reported in a specified journal, with a specified year range and name of author.

The results that are obtained by this search are presented as a list of polymers which have at least one sample matching the search terms. Each polymer has a link to the webpage of that polymer, as well as a button to a page containing only matching samples. The button also displays how many samples of that polymer matched your search.

The *advanced search* page adds a number of options for more precise searching. There is a box to specify a minimum and maximum *average*

Polymer	property points
polyethene	3264
polystyrene	2240
poly(methyl methacrylate)	1279
poly(ethylene oxide)	1520
poly(prop-1-ene)	2287
poly(vinyl chloride)	963
polyaniline	541
poly(ethylene terephthalate)	1706
poly(bisphenol A carbonate)	1043
polypyrrole	242
poly(hexano-6-lactam)	1108
poly(tetrafluoroethylene)	467
cis-1,4-polyisoprene	556
poly(dimethylsiloxane)	427
poly(vinylidene fluoride)	705
poly(vinyl acetate)	248
poly[(hexane-1,6-diamine)-alt-(adipic acid)]	504
poly(vinyl alcohol)	586
poly(propane-1,2-diol)	246
polyformaldehyde	404
poly(2-methylprop-1-ene)	100
polyetheretherketone(PEEK)	567
poly[(diaminodiphenylether)-alt-(pyromellitic anhydride)]	326
trans-polyacetylene	22
polyethersulfone(PSF)	225
polyacrylonitrile	417
polyacetylene	25
poly(2,6-dimethylphenol)	190
poly(p-phenylene sulfide)	378
poly(butyl methacrylate)	121

Table 4.2: Table of the *popular polymers* in the PoLyInfo database.

Polymer Basic Search

Help Japanese
Help English

- **Polymer or Material Name**
 Other
☒ substring ☐ complete
- **Material Type** (search with other search condition)
- **Polymer Type** (search with other search condition)
- **Polymer class**
- **CU Formula of Homopolymer or a Component** (☐ search polymers consist of selected atoms only)
C H - -
(ex. 3-5)
- **Property** (For properties with * in menu, range conditions are ignored)
 Range
min: max:
- **Reference**
Journal Other Year (ex. 1998 - 2000) -
Author (substring)

Figure 4.4: The basic search menu from the PoLyInfo database.

molecular weight. This average molecular weight is not a specific average, but instead if any of the average molecular weights of a sample match then it will match the search. This means that M_n and M_w will both match, for example. The *shape of test piece* can also be searched against. This is selected from a drop-down box with the options of film, sheet, powder, pellet, fibre, block, single crystal, solution, disk and cylinder. The degree of crystallinity can also be specified as a minimum and maximum, as well as an option to require the data to possess *crystallographic information*. The final difference for the advanced search is that now up to three properties can be specified, rather than the one allowed in the basic search.

The *structure search by parts* allows you to specify which fragments make up the CU in the polymer. The fragments vary from being as small as a methyl group to a large fused ring system containing eight aromatic rings.

4.4 Polymers

The database consists primarily of the following types of pages: polymers, samples of polymers and monomers. These will be discussed in the following

sections. A polymer means a specific repeat unit, for example poly(ethene), while a sample is a specific experimental sample which has been reported in the literature. Samples of the same polymer can have very different properties due to variation as described in Chapter 1.

4.5 Polymer page

From the PoLyInfo database 13,402 different polymers were added to PIKS. An example polymer page can be seen in Figure 4.5. The polymer pages detail a variety of information:

- IUPAC source and structure based names for the polymer
- Other common names, for example “kevlar”
- Polymer classifications, such as “polyolefin”
- Formula weight of the repeat unit
- Elemental formula of the repeat unit
- Predicted properties from group contributions
- Textual representations of the repeat unit structure

The IUPAC names are generated automatically by a subcomponent of the database (IUPAC nomenclature) as show in the architecture diagram (Figure 4.2). This naming does produce errors, however, which are discussed in Chapter 6. The structure-based names are calculated from the repeat unit information stored in the database. The source-based names are presumably calculated from the monomer information, but there are some inconsistencies which might imply they are entered manually.

A polymer can be a member of more than one class, so some polymers are members of two or three classes while the majority are members of only one. Properties such as glass transition temperature are calculated using a group contribution subsystem which means that each polymer will have one

polyaniline

PID: P460064

IUPAC structure based name:

poly(imino-1,4-phenylene)

poly(imino-p-phenylene)

IUPAC source based name:

polyaniline

Other name:

poly(aniline)

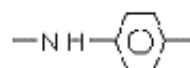
Polymer class: Polyimines

Formula weight(FW): 91.11064

CU formula: C₆H₅N

Text formula(TXF): -NH-{pC₆}-

CU Chemical Structure:



PoLyInfo formula: plain(133/n/{1,-,A1+} {2,-,0} 708/n/{1,-,0} {7,-,A1-})

[Summary of property data](#)

[Polymerization paths & Candidate monomers](#)

28 copolymers including this component

Figure 4.5: A typical polymer page from the PoLyInfo database.

property value predicted for it since M_w and M_n are not taken into account in the group contribution method of property prediction.

The textual representation of the repeat unit structure comes in two types; the TXF (text formula) and the PoLyInfo Formula. These are two separate systems for describing the repeat unit. The TXF formula was developed by the Zephyr Corporation and is a proprietary system. The TXF formula appears initially to be the more readable of the two representations, but is not able to represent complicated ring structures. For polymers containing such systems the TXF formula is wrong and simply omits the complex components from the structure. For simple repeat units the TXF is easily human readable. For example for polyethene the TXF is simply -CH₂-.

The PoLyInfo formula (PIF) is more complicated, but complete. For polyethene the PIF is :

```
plain(129/n/{1,-,A1+}{2,-,A1-})
```

This represents the same repeat unit as the previous TXF of -CH₂-. The PIF format allows for many more different fragments than the TXF formula, and can represent the repeat units of the entire database. Both these descriptions of the repeat unit appear, from the architecture diagram, to be generated from the same internal representation of the repeat unit in the database. There is no-way of accessing the internal data-structure that generates these representations.

4.6 The PoLyInfo Formula

The PoLyInfo formula (PIF) has little documentation on the PoLyInfo website. In order to convert this into PML it was necessary to understand how the format worked. The `plain()` is always present and surrounds the formula completely. Within the plain parentheses the formula is broken down into fragments. Each fragment has a number and one or more connection point data. The format of this is the fragment number followed by `/n/` (eg, 129/n/) followed by the connection point data. This connection point data consists of a set of curly braces with a number, comma, symbol for the bond order,

comma, and finally a 0, A1+ or A1-. The A1+ indicates the start of the repeat unit while the A1- indicates the end. The 0 is used for all other cases. In the case of polyethene the formula of `plain(129/n/{1,-,A1+}{2,-,A1-})` is then explained as fragment 129 joined at connection point 1 to the previous repeat unit with a single bond, and at connection point 2 joined to the next repeat unit with a single bond.

A more complicated example is polystyrene. Polystyrene has the PIF:

```
plain(130/n/{1,-,A1+}{2,-,0}{3,-,0}(708/n/{1,-,0})129/n/{1,-,0}{2,-,A1-})
```

As shown in Figure 4.6 this is broken down into three fragments. The first is `130/n/{1,-,A1+}{2,-,0}{3,-,0}`. The 130 fragment refers to a CH group. The three connection point data enclosed in “{}” are respectively the starting join, the exiting join and finally the side-chain. In CH all three connection points are attached to the carbon so the numbers 1, 2 and 3 refer to connections to the carbon atom.

The second is `(708/n/{1,-,0})`. The 708 fragment is a phenyl ring. This has only one connection point datum since it is joined onto the backbone, but has no continuing fragments from it. The enclosing “()” is used to denote a branch from the main chain.

Finally `129/n/{1,-,0}{2,-,A1-}`. The 129 fragment is a CH₂ group. This is joined from position 1 to the previous fragment (the CH group) and from position 2 to the next repeat unit along.

For fragments which have connection points on more than one atom, it is important to know which numbers correspond to which atoms. The PoLyInfo database has pictures of fragments with their fragment number, but does not include the connection point information. These pictures were manually converted into CML fragments. In order to get the connection point numbers it was therefore necessary to search the database for examples of the fragments being used. In cases where there were a large number of different connection points it was necessary to find multiple examples of polymers which included the fragment in order to unambiguously identify the connection point numbering.

The numbering scheme is not immediately intuitive. In the case of Kevlar (Figure 4.7) the PoLyInfo formula is :

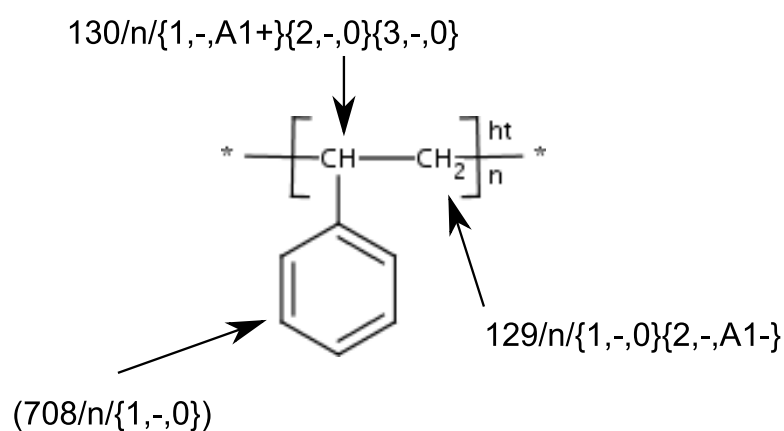


Figure 4.6: How the PoLyInfo formula is structured. The 130/n/ refers to the left hand CH group. The {1,-,A1+} indicates that connection point 1 of the CH group is bonded by a single bond to the previous repeat unit. The {2,-,0}{3,-,0} indicates that there are two further fragments bonded to the CH group, using connection points 2 and 3. The (708/n/{1,-,0}) is a phenyl ring connected to the 3 position on the CH group, using position 1 on itself. The 129/n/{1,-,0}{2,-,A1-} represents a CH₂ group connected to the CH group by the position 1 on the CH₂ group and position 2 on the CH group. The CH₂ group is also joined to the subsequent repeat unit on position 2 of the CH₂ unit and position 1 of the next CH group.

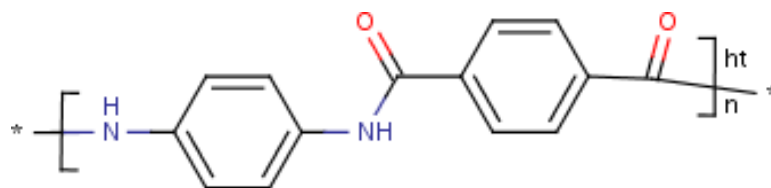


Figure 4.7: The structure of kevlar.

```
plain(133/n/{1,-,A1+}{2,-,0}708/n/{1,-,0}{7,-,0}133/n/{1,-,0}{2,-,0}
207/n/{1,-,0}{2,-,0} 708/n/{1,-,0}{7,-,0}207/n/{1,-,0}{2,-,A1-})
```

The initial $133/n/\{1,-,A1+\}\{2,-,0\}$ corresponds to the NH group at the start. This is joined to the previous repeat unit through connection point 1, and to the next fragment through connection point 2. This next fragment is $708/n/\{1,-,0\}\{7,-,0\}$, which is a phenyl. This is connected by point 1 to the previous fragment, and point 7 to the next fragment. $133/n/\{1,-,0\}\{2,-,0\}$ is another NH group, connected to the previous and next fragments through points 1 and 2. $207/n/\{1,-,0\}\{2,-,0\}$ is a C=O fragment, which is connected through points 1 and 2 (both on the carbon) to the previous and following fragments. $708/n/\{1,-,0\}\{7,-,0\}$ is another benzene connected at points 1 and 7. The final fragment, $207/n/\{1,-,0\}\{2,-,A1-\}$ is another C=O fragment joined to the previous fragment and the following repeat unit through the carbon atom.

The first puzzle is the use of the number 7 to mean a 1,4 or para benzene connection. Further investigation reveals that the vast majority of atoms each have two connection point numbers, irrespective of whether that makes chemical sense. This means that a benzene will use the connection point numbers of 1,3,5,7,9 and 11 instead of the IUPAC 1,2,3,4,5 and 6 numbering. This is not always the case, however, as will be shown in the following example.

For fused ring fragments, the numbering scheme is slightly different. In the example of poly(oxy-1,4-phenylenesulfonylnaphthalene-2,7-diylsulfonyl-1,4-phenylene) (Figure 4.8) the PoLyInfo formula is given by:

```
plain(106/n/{1,-,A1+}{2,-,0} 708/n/{1,-,0}{7,-,0} 302/n/{4,-,0}{5,-,0}
1304/n/{4,-,0}{15,-,0} 302/n/{4,-,0}{5,-,0} 708/n/{1,-,0}{7,-,A1-})
```

Things to note are that the O=S=O fragment is given by $302/n/\{4,-,0\}\{5,-,0\}$

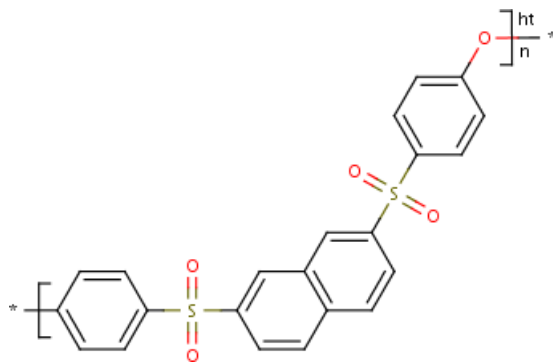


Figure 4.8: The structure of poly(oxy-1,4-phenylenesulfonylnaphthalene-2,7-diylsulfonyl-1,4-phenylene)

and uses connection point numbers 4 and 5 even though there are no other plausible connection points in the fragment. The fused ring is given by 1304/n/{4,-,0}{15,-,0}. If each carbon had two numbers, as with phenyl, then both these connection points would be odd. The only explanation that fits with the other examples is that the carbon atoms at the ring junctions have only one number assigned to them. The numbering then starts at one of these carbons with 1. Numbering then progresses anti-clockwise around the ring with two numbers being assigned to each carbon (but only the first of these two numbers being used) making the carbons 2+3, 4+5, 6+7, 8+9, 10, 11+12, 13+14, 15+16, 17+18. The two connections at numbers 4 and 15 are therefore explained.

A program was written to convert the PoLyInfo formula automatically into PML. This is described in Chapter 5, Section 5.2.4. This was then later used in Chapter 6 to validate the data in the PoLyInfo database by comparing the reported PoLyInfo formula with the other data given for consistency.

4.7 Samples of Polymers

Each polymer can have any number of samples of that polymer. From the PoLyInfo database 69,465 samples were added to PIKS. Each sample refers to an actual unique physical sample of that polymer which has been reported in the literature. Each sample can possess both data describing the physi-

cal properties of that sample, as well as characterisation data of the sample. There is no requirement for characterisation data of the sample of the polymer to be recorded, but a large number of samples do have the M_w and M_n molecular weight averages recorded. Other characterisation data that are sometimes present are the crystallinity, the degree of branching, method of polymerisation used and the M_z and M_v molecular weight averages. An example of a page for a sample is shown in Figure 4.9. This sample is of a polystyrene resin made by the Dow Chemical Company with 43% crystallinity measured by DSC (differential scanning calorimetry). Other samples could contain additional information while omitting some of the fields present in this sample.

4.8 Properties

Each sample of a polymer has its own set of physical data properties. The samples in PIKS from the PoLyInfo database contain 262,396 property points. In Figure 4.9 the bottom half of the page contains the physical property data points recorded for that sample. This particular sample of polystyrene has a glass transition temperature, a melting temperature and a heat of fusion measurement reported. The property information also reveals the conditions and method used to measure the property. In these cases DSC was used with a heating rate of 20K / min.

Half of samples in the PoLyInfo database have either one or two properties. In Figure 4.10 the distribution of properties per sample is shown. The properties cover a range of different types of physical property types. These properties are grouped into seventeen different classes of property (Table 4.3).

4.9 Physical Properties

The only property in the physical property class is *density*. The density is the mass per unit volume of a substance. This property can have a subproperty of specific volume, which is the inverse of the density. The database defines

Sample Information

SampleID: 00292-1-1-1

Name: polystyrene

Material type: neat resin

Polymer class: Polystyrenes , Polyvinyls

Trade name/ Grade/ Manufacturer: // Dow Chemical Company

Stereoregularity: st

Processing info.

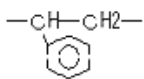
- Injection
- As received

Crystallinity: 43 [%](DSC)

Formula weight(FW): 104.14912

CU formula: C₈H₈

Text formula(TXF): -CH({C6})-CH₂-

CU Chemical Structure: 

Reference: Beckmann, Jorg; McKenna, Gregory B.; Banks, David H.; Landes, J.
Plastics Engineers , 54th , 2 , 2272-2276(1996)

Properties

- Glass transition temp.
 - Glass transition temp.: 95 [C]
 - Method: DSC
 - Condition: Heating rate; 20K/min
- Melting temp.
 - Melting temp.: ca 275 [C]
 - Method: DSC
 - Condition: Heating rate; 20K/min
- Heat of fusion
 - Heat of fusion: 53.2 [J/g]
 - Method: DSC
 - Condition: Heating rate; 20K/min

Related Information

[Other polymers in this literature](#)

Figure 4.9: An example sample page. In this case the characterisation data for the sample contain the processing information, the crystallinity, stereoregularity and the name of the manufacturer.

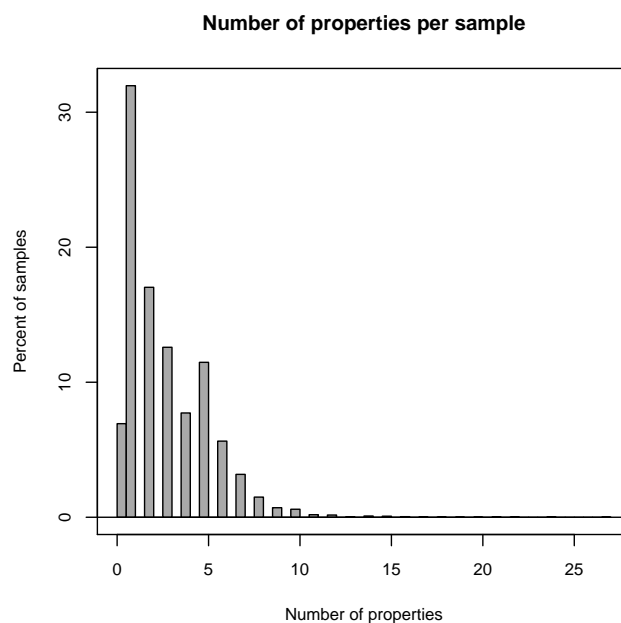


Figure 4.10: The distribution of number of reported properties per sample. About half of the samples have either one or two properties reported.

Physical properties	Flexural properties
Optical properties	Compression characteristic
Thermal properties	Creep characteristic
Electrical properties	Heat characteristic
Physicochemical properties	Impact strength
Dilute solution properties	Hardness
Rheological properties	Heat resistance and Combustion
Tensile properties	Other physical properties
Shear properties	

Table 4.3: Table of the different property classes in the PoLyInfo database.

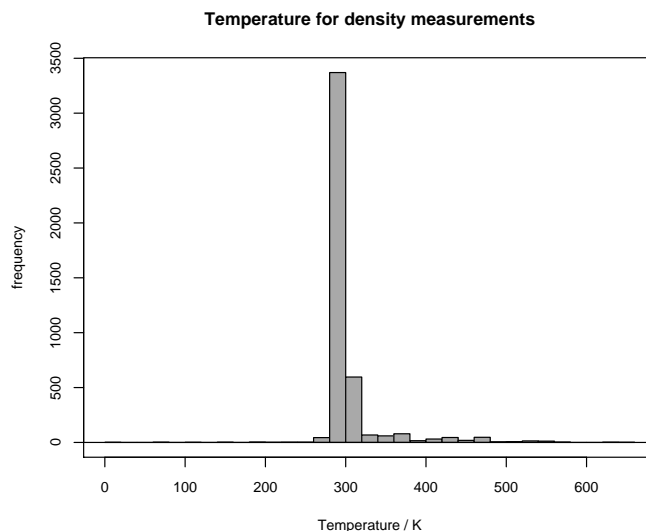


Figure 4.11: The distribution of temperatures at which density measurements were made.

the temperature for all density measurements should be taken at to be 23 °C. However, since the data is taken from the literature when the temperature is often not stated, this precondition is mostly meaningless. There are 9,093 density measurements, with 4,432 (48.7%) having a reported temperature condition. The majority of these reported measurements were made near 23 °C, but as can be seen in Figure 4.11 there is some variation.

4.10 Optical Properties

The optical properties in the database are: *refractive index* and *stress-optical coefficient*. The refractive index is the ratio of the speed of light in a vacuum to the speed of light in the sample. The temperature and the wavelength of light used affect the results, and these are both included in the database as preconditions if they are reported in the literature. There are 2,338 measurements of refractive index, with 1,916 (82%) of these having some conditions reported. 896 (38%) of the measurements have a temperature recorded, and 1,013 (43%) have a defined wavelength. Some of the entries have a source

specified, e.g. 10 list *sodium lamp*, but no specified wavelength.

The stress-optical coefficient is used in stress measurements. Light entering an anisotropic substance can experience birefringence, as the refractive index depends on the polarisation of the light, which splits the path of the light. Macromolecules in a polymer can align due to an applied stress, producing anisotropy. This then causes a degree of birefringence which depends on the applied stress, and the *stress-optical coefficient*. The stress-optical coefficient is a measure of how much birefringence, due to anisotropy of the substance, is introduced for a given stress. The temperature and wavelength of light are recorded as preconditions if available. There are 266 measurements of the stress-optical coefficient, with 211 (79%) of these having some condition specified. Of these, 166 (62%) have a specified temperature and 17 (6.4%) have a specified wavelength.

4.11 Thermal Properties

In this class we have glass transition temperature, melting temperature, heat of fusion, crystallization temperature, heat of crystallization, crystallization kinetics, LC phase transition temperature, thermal decomposition temperature, isothermal weight loss, linear expansion coefficient, volume expansion coefficient, thermal conductivity, thermal diffusivity, specific heat capacity (C_p) and specific heat capacity (C_v).

Out of these the glass transition temperature (T_g) and melting temperature (T_m) are the most commonly reported properties. The melting temperature is the temperature at which the crystalline components of a polymer melt. The melting temperature can be measured using a range of techniques including a Differential Scanning Calorimetry (DSC) experiment. In this the amount of energy input required to increase the temperature of a sample is measured. The melting temperature appears as a maximum in the plot of heating rate against temperature. The glass transition temperature, however, is a change in the heat capacity of the polymer rather than a phase change. As such, it appears as an incline in the DSC graph rather than a minimum or maximum. This means that the glass transition happens over

a range of temperatures and so the measurement is not as precise as the melting temperature. Glass transition temperatures can also be measured by a range of other methods such as Dynamic Mechanical Analysis (DMA), which measures the change in mechanical properties with temperature.

The thermal decomposition temperature and isothermal weight loss properties are both three dimensional properties. They include a temperature, a weight loss percent and a time. Each sample which has these properties recorded often has two or more measurements at different temperatures or times. Since these properties are dependent on multiple variables for one sample, the limited number of data points makes the data sparse and difficult to predict.

The LC (liquid crystal) phase transition temperature is the temperature at which a phase change has occurred. The property has a remark field which, in some cases, specifies which two phases are being transitioned between. The language used in this field is free text, which makes parsing complicated.

The heat of fusion property is the energy released when the molten polymer solidifies. This property is, according to the PoLyInfo documentation, in both Energy / Mass, and Energy / mol. However in the database there are some entries with values in Energy / Volume as well. This causes problems with comparisons since the density of that specific sample is not always available, and is required to convert between Energy / Mass and Energy / Volume.

4.12 Electrical Properties

The electrical properties referred to in PoLyInfo include dielectric constant, dielectric dispersion, conductivity and dielectric breakdown voltage. The dielectric constant is a measure of the polarisation between two charges when this medium is subjected to an electric field. A vacuum has a dielectric constant of 1. All substances have a positive dielectric constant. The dielectric constant is dependant on temperature and frequency. There are 2,575 dielectric constant measurements, with 1,391 of these having both a temperature and a frequency condition. The distribution of temperatures is shown in

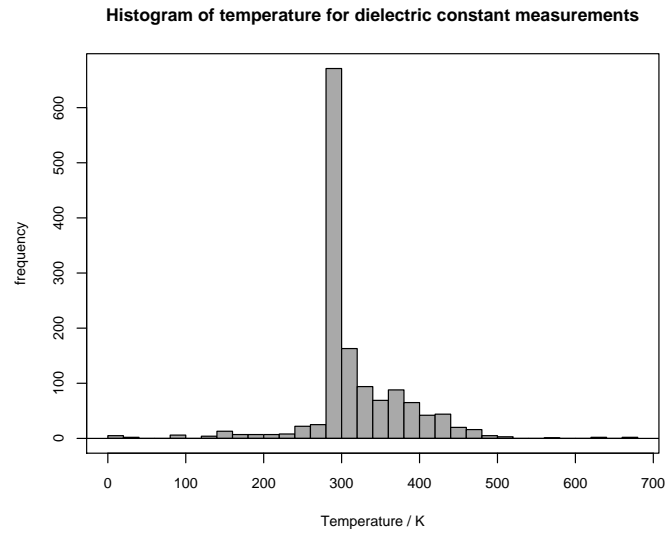
Figure 4.12(a), the distribution of frequencies in Figure 4.12(b). The temperature distribution shows that the majority of measurements are done at room temperature, with more conducted at raised rather than lowered temperatures. The frequency measurements were mostly done at 1 kHz, with a spread at lower and higher frequencies.

The conductivity depends on the temperature. The spread of temperatures recorded are shown in Figure 4.13. As with the dielectric constant the majority of measurements were made at room temperature, with more measurements made at an elevated temperature than a reduced temperature.

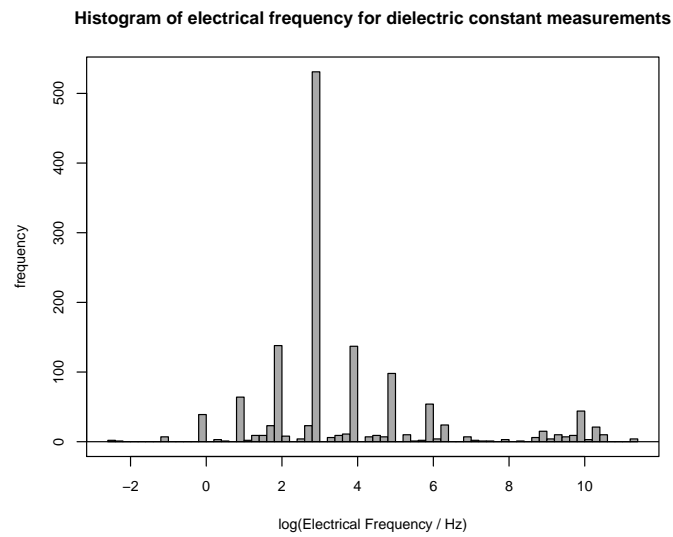
4.13 Physicochemical Properties

This category of properties includes the solubility parameter, cohesive energy density, water absorption, water vapour transmission, surface tension and interfacial tension. The solubility parameter is particularly important as it allows predictions to be made on the solubility of a polymer in different solvents. The closer the solubility parameters of the polymer and the solvent the more soluble the polymer will be. The presence of crystallinity, cross-linking and higher molecular weights will lower the solubility. This solubility parameter property is the Hildebrand solubility parameter[48] which is the square root of the cohesive energy density. This means that no account of polarity or hydrogen bonding is included. The PoLyInfo database also has some measurements of the Hansen solubility parameters[49]. These are three parameters which account for the effects of dispersive, polar and hydrogen bonds. They are called δ_d , δ_p and δ_h respectively. In 1092 samples the solubility parameter is reported. In 516 (47%) of these cases a condition was recorded. 452 (41%) of these had a temperature condition, and 105 (9.6%) had a solvent reported. In 125 (11%) samples the Hansen parameters were also recorded.

The water absorption property is the percent of weight of water absorbed under certain conditions. These conditions range from the time, immersion in water (or relative humidity of the air if not immersed) as well as the



(a) Temperature conditions.



(b) Frequency conditions

Figure 4.12: Conditions of dielectric constant measurements.

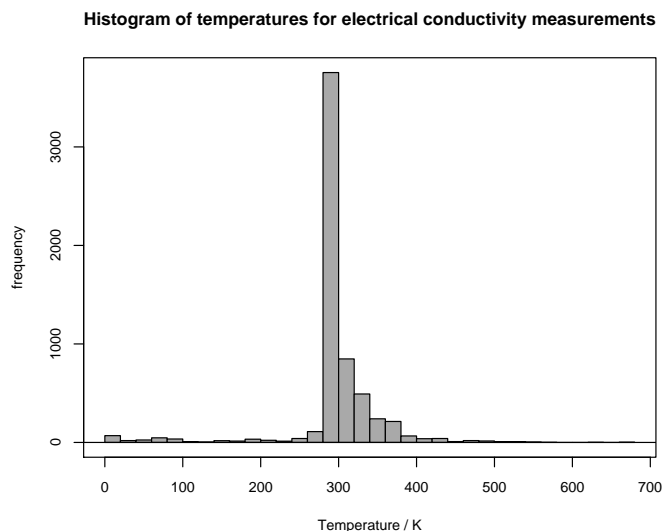


Figure 4.13: The distribution of temperatures at which a conductivity measurement was made.

temperature. Out of the 990 samples with a water absorption property, 933 have at least one condition recorded.

4.14 Dilute solution properties

The dilute solution properties include the radius of gyration and the intrinsic viscosity. The radius of gyration is a measure of the size of the macromolecules in the sample from light-scattering. The intrinsic viscosity is the limit of the viscosity when the concentration of macromolecules falls to zero. This number is related to the molecular weight of the macromolecule by the Mark Houwink[8] equation: $[\eta] = KM^\alpha$ where α depends on the solvent and polymer used. α takes the value of 0.5 for a theta solvent, with larger values for better solvents. There are 6417 samples with a intrinsic viscosity measurement. Out of these 6241 (97%) have some condition, 5550 (86%) have a temperature recorded and 6107 (95%) have a solvent recorded.

4.15 Monomers

The PoLyInfo database contains 16,371 monomers. These monomers are divided into 38 different monomer classes based on chemical structure. These classes are shown in Table 4.4 along with the number of monomers in each class. These classes are not exclusive, so a monomer can be a member of more than one class.

Each monomer has an IUPAC name, a unique PoLyInfo monomer ID, in some cases a Chemical Abstracts Service (CAS) number and JST substance number. If a monomer's structure does not fit any classifications, it is classified into the "Other_monomers" class.

The monomers are described by their SMILES to give their connection table. A molecular formula and a molecular weight are also given. The molecular weight is calculated from the molecular formula while the SMILES string is independent of the molecular formula (See Chapter 6).

Monomers also have a 2D and 3D picture representation which has been generated from the SMILES string. There is also a link to a list of polymers that can be made from this monomer. This list of polymers also specifies which other monomers (if any) are required to make a particular polymer from this monomer.

4.15.1 JST

The Japan Science and Technology Agency (JST) host a database¹ of small molecules. As the PoLyInfo database monomer entries contain JST numbers the use of the JST database in conjunction with the PoLyInfo database allows data to be checked from multiple sources for validation. This is described in Chapter 6. Not all the monomers in the PoLyInfo database contain JST numbers but 12600 (77%) monomers out of 16371 do, with 12532 distinct JST numbers.

¹http://nikkajiweb.jst.go.jp/nikkaji_web/pages/top_e.html

Monomer Class	No. Monomers	Monomer Class	No. Monomers
Vinyl compds	5255	Cyclic olefines	162
Acrylic acids	2885	Halo-olefins	160
Dicarboxylic acids	1953	Lactams	157
Diols	1850	Lactones	145
Diamines	1496	Phosphorus contg compds	121
Phenols	1268	Cyclic acid anhydrides	117
Styrenes	1106	Aldehydes	117
Dienes	906	Hydroxy acids	112
Other monomers	687	Cyclic imides	89
Dihalides	628	Silicon contg cyclic compds	86
Sulfur contg compds	467	Cyclic iminoethers	80
Cyclic ethers	444	Melamines and Ureas	79
Diisocyanates	347	Silane compds	75
Cyclic sulfides	321	N-carboxy anhydrides	57
Acetylenes	320	Carbonates	55
Anilines	268	Phosphorus contg cyclic compds	46
Olefins	259	Diketone	18
Amino acids	212	Cyclic carbonates	15
Cyclic amines	163	Arom ethers	13

Table 4.4: Table of the different monomer classes along with their number of entires as defined in the PoLyInfo database.

4.16 Problematic data

Some of the data in the PoLyInfo database was immediately problematic. The reasons for this are outlined below.

4.16.1 Kelvin vs Celsius

By inspection of the glass transition temperature data, there is an obvious outlier at 48K. As the PoLyInfo database maintains provenance, the paper responsible for the datapoint can be checked to see if this is a transcription error or a valid discovery. The reference given by the PoLyInfo database is

“Fytas, George , Physical Optics of Dynamic Phenomena and Processes in Macromolecular Systems , 205-215 (1985)”

which does not appear to exist, however George Fytas did present work at the 1984 conference[50]. Through private correspondence with George Fytas it was confirmed that the datapoint should be 48C, not 48K. The data was therefore excluded from the analysis.

4.16.2 Polydispersity

As mentioned in section 1.1.3 the polydispersity (M_w/M_n) normally has values of between 1 and 3 depending on the polymerisation method used. There are a few datapoints in the PoLyInfo database, all from the same paper[51], with wildly different values. For example, one sample has an $M_n = 11700$ and $M_w = 6500000$ giving a polydispersity $M_w/M_n = 556$. These values lie far outside the rest of the samples in the database and so would cause problems for analysis.

The paper reveals that the M_w and M_n values are not, however, values of an actual polymer sample. Instead the values are that of mixtures created by mixing a high molecular weight and a low molecular weight sample together. This means the M_w/M_n ratio is no longer indicative of the width of the distribution of molecular weights present in the sample, as the combined sample is the sum of two different distributions and so the usual metric for

the width of a distribution is meaningless in this case. For this reason these samples were excluded from the analysis.

4.17 Aggregate data contained within PoLy-Info

Empirically it is found that the glass transition temperature of a polymer and its melting point are correlated by a simple relationship: [46, 52, 53, 54]

$$T_g \approx \frac{2}{3}T_m$$

Using the data in the PoLyInfo database it is possible to test if this empirical relationship holds when using considerably more data than the original studies which used up to 132 data points. Data was selected from all the samples of polymers which did not contain additives, and had both a glass transition temperature and a melting point measured. Out of the 54,910 samples which did not contain additives 3,414 contained both a T_g and T_m , distributed across 1,261 different polymers. These data were then used to produce a scatter plot of T_g vs T_m and the regression line calculated using the package R[55]. (Figure 4.14). This gives a gradient of 0.700 with an r^2 of 0.98. The reason the r^2 value is so high is that in a linear regression with no y-intercept the calculation of r^2 is done with the usual \bar{y} term is replaced by 0 [56]. If a y-intercept is allowed then the gradient is 0.658 with a y-intercept of 20.1K and an r^2 of 0.69.

This shows the approximate relationship identified with 132 data points still holds up with 3,414 datapoints from 1,261 different polymers. A better model is (approximating 0.658 with 2/3):

$$T_g = \frac{2}{3}T_m + 20$$

or

$$T_g = 0.7T_m$$

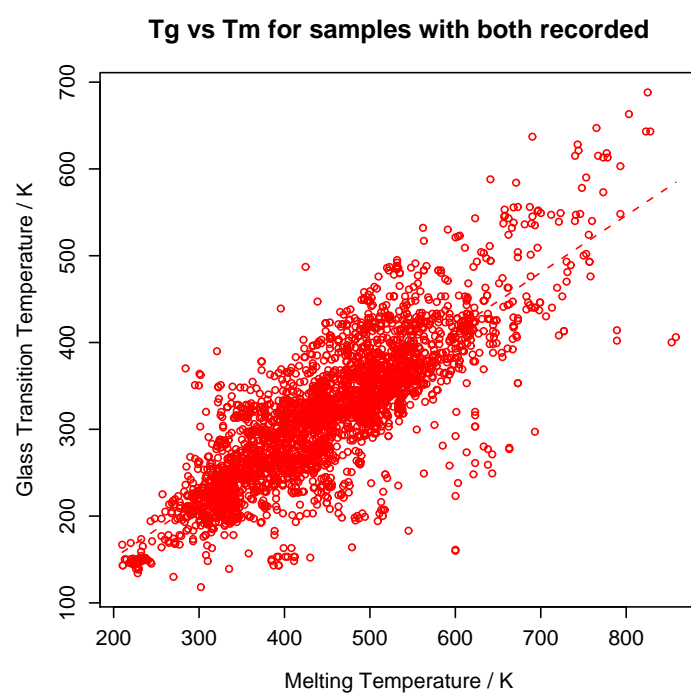


Figure 4.14: Scatter plot of T_g vs T_m in PoLyInfo database

One interesting part of the data in the PoLyInfo database is the existence of a handful of datapoints with a higher glass transition temperature than the melting temperature. In these cases the melting temperature was due to the melting of polymer side chains, while the glass transition temperature was due to the polymer backbone with another phase transition at third higher temperature[57]. If these datapoints are removed, the models before remain identical to 2 decimal places, other than y-intercept which is lowered to 15.6K and larger r^2 values. Since these materials have more than 2 phase transitions, it could be argued that they should be excluded from the dataset. Since the value of the y-intercept can be changed considerably by the removal of 20 data points out of 3,414 it makes the linear regression with no y-intercept is a better choice since it is more robust with respect to small changes in the dataset.

4.18 Conclusions

The PoLyinfo database is a useful public repository of polymer data. The documentation describing the units properties are recorded in, and whether a condition is a requirement for a property data point, are not adhered to in the database itself. When the condition is not recorded this can render the data useless. For example the intrinsic viscosity depends strongly on the solvent used and a value is essentially meaningless if listed without the solvent. The lack of compliance with the stated policies shows that the policies are not enforced automatically by the database.

Aggregating the data contained in the PoLyInfo database for glass transition temperature and melting temperature of samples containing both measurements showed that the aggregate data in the PoLyInfo database is comparable with data from 1952. The previous relationship between T_g and T_m , which was found using only one value per polymer, also holds true when used on samples of polymers of the much larger modern dataset.

The data from the PoLyInfo database was used in Chapter 6 for automatic validation, and in Chapter 7 for machine learning. The tools used to extract, process and analyse the data in the PoLyInfo database are described

in Chapter 5.

Chapter 5

Polymer Informatics Knowledge System

5.1 Objectives

This chapter describes the Polymer Informatics Knowledge System (PIKS), which has been developed in order to enable the automated extraction of polymer data from publicly available sources, to convert it into an extensible data structure, and to build tools to analyse, validate and model the resulting corpus. These tools were used to extract the data which was used in Chapter 4. The data source of the PoLyInfo database was chosen due to its freely accessible nature and the depth of information contained within. There are other polymer databases, but they either charge for access[42, 43], or contain only very limited amounts of data[44, 45]

Use of the validation component of PIKS is discussed in Chapter 6 and of the modelling component in Chapter 7.

5.2 Code for extraction and parsing of PoLy-Info

In order to investigate the data in the PoLyInfo database, it was necessary to produce a range of tools to obtain, parse and convert the data held within.

The following Java components of PIKS were used to parse and download data.

5.2.1 Polyinfo-harvester

This is represented in Figure 5.1 by the *PoLyInfo-Harvester* box. This project has the main class of **Harvester**. Running this class from eclipse or invoked from the command line will then cause the spider to download the publicly accessible PoLyInfo website and save into a subfolder called “database”. If the spider is interrupted, it will resume from where it left off when it is run again using a log file. It downloads the polymer pages, the sample list of each polymer, and then the individual sample pages themselves. It also downloads the monomer pages. In order to avoid flooding the PoLyInfo database with requests, the harvester uses a few second delay between each request. The harvester produced 224,254 files over the course of a month. An initial partial download took place in 2006, with the final complete download taking place in November 2008. This project was initially written by Nick Day, with some modifications made myself. Since 2008 the PoLyInfo website has undergone a redesign which would require significant additional work to accommodate in the spider and parsing tools. Since the website has only added a small quantity of additional data since then this has not been carried out.

The HTML from the PoLyInfo database is sometimes badly-formed. Before it can be parsed by XML[13] tools, the downloaded HTML is parsed through TagSoup[58] before being saved as XHTML. TagSoup parses “wild” HTML and fixes unclosed tags, overlapping tags and other problems. For example many HTML documents do not close their <head> or <body> element. As valid HTML is not necessarily valid XML, this step would be necessary even if the HTML were well-formed.

5.2.2 Property

This project contains the bulk of the PoLyInfo parsing code. The main class **SamplePropertyExtractor** (shown as *Property Sample* in Figure 5.1) parses

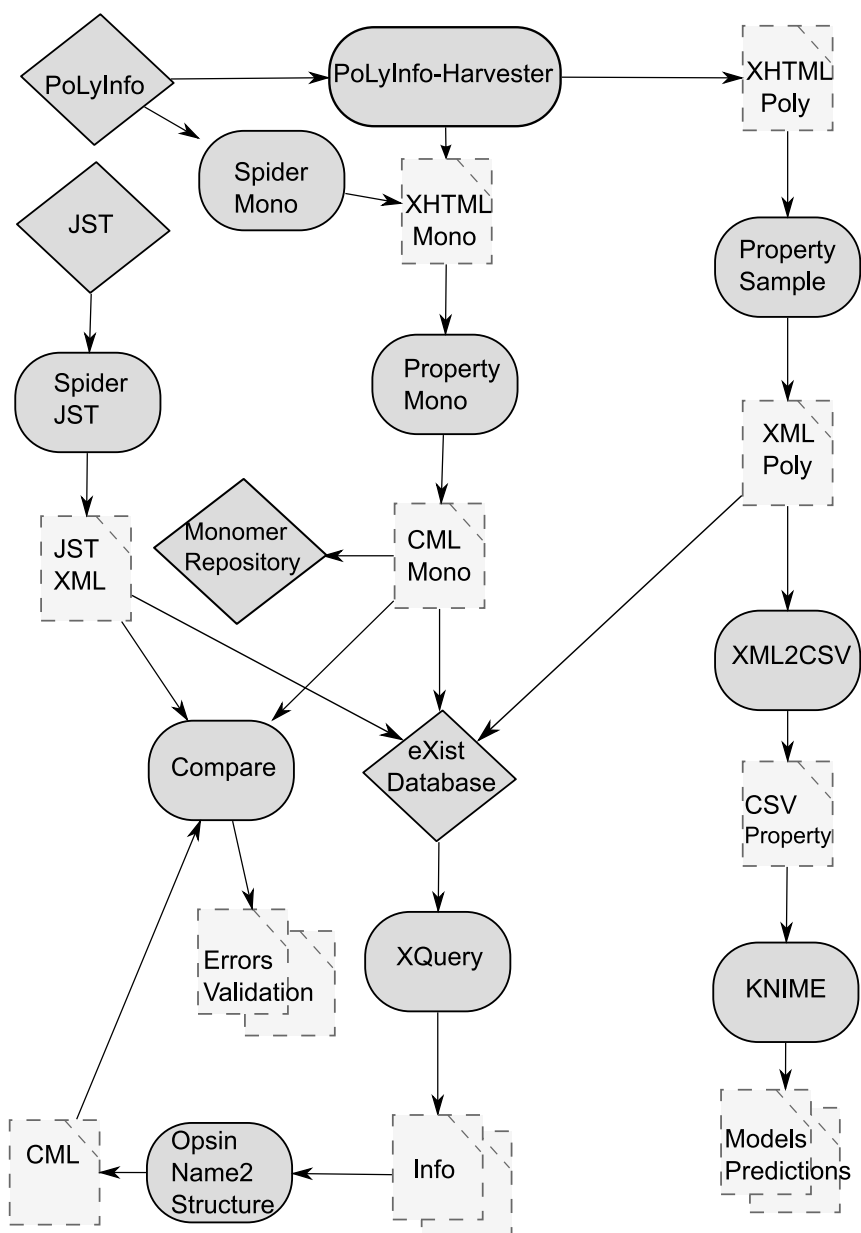


Figure 5.1: The different tools and code used to download and extract information from the PoLyInfo Database. Not everything is included in this diagram to avoid over complication. The diamonds represent data repositories, the rectangles with rounded corners represent Java projects and the squares with the dashed lines represent files. The components presented here are described in the following pages.

all the XHTML files which were downloaded by the Polyinfo-harvester and produces one XML output file per polymer, containing all the information from the samples of that polymer. This is achieved by parsing each polymer file, then all of the sample files for that polymer, adding XML nodes to a document as it progresses. This XML tree is then saved into a file as the output. Units are parsed where possible and stored as XML attributes. The meta-data for each sample is also saved, including the reference to the paper it was published in.

The syntax of the XML file is a `<Polymer>` element as the root of the file. This has `id`, `name` and `class` attributes, with values from the database. The `<Polymer>` element has any number of `<Sample>` element children. These child elements contain the information from a particular sample.

Each `<Sample>` contains an `id` attribute with a unique sample ID. There are then a range of child properties depending on the data available from the database for that sample. A `<Reference>` element is always present, containing the text of the reference where the sample was found. Sometimes the sample contained additives, and if this is mentioned in the database an `<Additives>` element is present containing the text string description of the additive used. If the polymerisation method was mentioned, a `<PolymerisationMethod>` element records this information.

Other data from the sample is contained in `<Property>` elements. This has an `id` attribute containing the name of the property, a `unit` attribute containing the units, and a `value` attribute containing the numeric value. A lot of properties are dependent on other variables, and if conditions are present in the database these are added as additional `<Data>` elements as a child of the `<Property>` element. When a property has a clear value, this has been parsed and stored as the `value` attribute. Some properties such as Thermal Decomposition Temperature have both a percentage weight loss, a temperature and a duration. This makes choosing a single value to represent the datapoint impossible. The data will be recorded as three `<Data>` elements attached to the parent `<Property>` element, which will have a blank `value` attribute. A single sample will often have several of these `<Property>` elements as they often have a weight loss percentage at

```

1 <Polymer id="P010002" name="poly(prop-1-ene)" class="Polyolefins">
2   <name convention="IUPAC structure based name">poly(1-methylethylene)</name>
3   <Sample id="13367-1-1-4">
4     <Reference year="1993">Taraiya, A. K.; Orchard, G. A. J.; Ward, I. M. ,
5       Plastics, Rubber and Composites Processing and Applications,
6       19 , 5 , 273-278(1993)</Reference>
7     <Property id="year" value="1993"/>
8     <Property id="Density" unit="[g/cm3]" value="0.9101">
9       <Data id="Specific volume">1.09878 [cm3/g]</Data>
10      <Data id="Method">Graded density column</Data>
11      <Data id="Condition">Mixture of propan-2-ol and digol</Data>
12    </Property>
13    <Property id="Refractive index" unit="[]" value="1.5128">
14      <Data id="Method">Abbe</Data>
15      <Data id="Condition">alpha-bromonaphthalene as contact liquid</Data>
16      <Data id="Remark">nM</Data>
17    </Property>
18  </Sample>
19  ...
20 </Polymer>

```

Table 5.1: An example truncated `<Polymer>` element. This example contains only one `<Sample>` element while the actual data contains 1768 samples of poly(prop-1-ene).

multiple times and / or temperatures.

An example of the data in the XML is shown in Table 5.1. Here a sample of poly(prop-1-ene) has its density and refractive index recorded. The full entry for this polymer has 1768 samples.

Monomer parsing

The class `Monomer` (shown as *Property Mono* in Figure 5.1) reads the eXtended Hyper-Text Markup Language (XHTML)[59] files which were downloaded from the PoLyInfo database for the monomers from both the polyinfo-harvester and the spider (Section 5.2.3) projects. These are then parsed to produce a CML XML file for each monomer. This contains both the structure of the molecule and the polymerisation data for which polymers can be synthesised using the monomer. The CML files are then added to an eXist[60] database for querying. An example monomer XML file is shown in Table 5.2.

The SMILES string from the PoLyInfo database is converted into a CML file using JUMBO converters. The additional information about the

```

1 <cml:molecule convention="PoLyInfo" id="M0101026" xmlns:cml="http://www.xml-cml.org/schema">
2   <cml:metadataList>
3     <cml:metadata name="dc:source">
4       http://www.polymerinformatics.com/data/polyinfo</cml:metadata>
5   </cml:metadataList>
6   <cml:name>1-propen-3-ol</cml:name>
7   <cml:name dictRef="cml:synonym">allyl alcohol</cml:name>
8   <cml:identifier convention="Polyinfo_Monomer_ID" value="M0101026"/>
9   <cml:identifier convention="CAS" value="107-18-6"/>
10  <cml:identifier convention="JST_Substance_No." value="J4.059B"/>
11  <classMembership>Olefins</classMembership>
12  <cml:formula inline="C3H6O"/>
13  <cml:property dictRef="cml:molarMass">
14    <cml:scalar dataType="xsd:double" dictRef="cml:molarMass" units="cml:dalton">
15      58.079
16    </cml:scalar>
17  </cml:property>
18  <cml:formula inline="OCC=C" convention="SMILES"/>
19  <monomerData>
20    <polymer id="P332048">poly(allyl alcohol)</polymer>
21    <type>Addition polymerization</type>
22  </monomerData>
23  <monomerData>
24    <polymer id="P440016">poly[(1-propen-3-ol)-alt-(furan-2,5-dione)]</polymer>
25    <type>Addition polymerization</type>
26    <coMonomer id="M1401001">furan-2,5-dione</coMonomer>
27  </monomerData>
28 </cml:molecule>

```

Table 5.2: Example XML for a monomer in PIKS eXist database.

monomer is then added to this CML file as additional XML elements. Since CML is an extensible format, these additional XML elements do not inhibit or interfere with CML processing tools. Molecular Operating Environment (MOE) [61] descriptors were calculated for the monomers and added to the CML as a demonstration of adding additional data to CML.

5.2.3 Spider

In Figure 5.1 this class is represented by the *Spider Mono* box. This project downloads some additional monomer data which the main harvester did not retrieve. This comprises of the polymerisation data about the monomers which lists which polymers each monomer has been used to create. This is run from the `PolyinfoMonomerDownloader` class. When run the spider will automatically parse a list of monomer IDs (generated from an XQuery of the PIKS current database, see Section 5.5) and retrieve the polymerisation

```

1 <Monomer id="M0101001">
2   <Property id="Nikkaji No.">J1.939I</Property>
3   <Property id="MF">C2H4</Property>
4   <Property id="MW">28.054</Property>
5   <Property id="CAS No.">74-85-1</Property>
6   <Property id="Legal No.">(2)-12 UN-1038 KU5340000 UN-1962
   ENCS2008153 TSCA (74-85-1)</Property>
7   <Property id="Names">
8     <name>Elayl</name>
9     <name>Ethene</name>
10    <name>Ethylene</name>
11    <name>Olefiant gas</name>
12    <name>Acetene</name>
13  </Property>
14  <Property id="Component"> - </Property>
15  <Property id="Use For">plant growth regulator</Property>
16 </Monomer>

```

Table 5.3: Example XML for JST data in PIKS eXist database.

data for each one. To avoid being a nuisance to the PoLyInfo servers time-delays between requests are implemented. The HTML downloaded is parsed through TagSoup to be converted into XHTML before being saved to disk.

In addition there is a class to download monomer files from the Japan Science and Technology Agency (JST) database¹, called **JSTResolver** (*Spider JST* in Figure 5.1). This class reads in a list of JST numbers and PoLy-Info Monomer ID numbers which has been produced from an XQuery on the PIKS eXist database (see Section 5.5). It then downloads each of the JST files to disk with a time delay to avoid overloading the JST servers.

The **testJSTDownload** class (also represented by *Spider JST* in Figure 5.1) then parses the downloaded HTML through TagSoup into XHTML. These intermediate files are then converted into XML files which are stored in the eXist database. These XML files allow the data present from the web-pages to be easily accessed using XQuery / XPath. An example of the XML produced is shown in Table 5.3.

5.2.4 PolyinfoReader

This project (not shown in Figure 5.1) is used to parse the PoLyInfo formulae (See Section 4.6) which have been extracted from previous programs,

¹http://nikkajiweb.jst.go.jp/nikkaji_web/pages/top_e.html

```

1 formula : "plain(" (fragmentjoin)+ ")"
2 fragmentjoin : fragmentid "/"n/" start (end)? (side "(" (fragmentjoin)+ ")" )*
3 fragmentid : (D|D D|D D D)
4 D : "0"|"1"|"2"|"3"|"4"|"5"|"6"|"7"|"8"|"9"
5 start : INITIAL_CONNECTION | CONNECTION
6 end : FINAL_CONNECTION | CONNECTION
7 side : CONNECTION
8 INITIAL_CONNECTION : "{" D+ "," ( "-" | "=" ) ",A1+}"
9 FINAL_CONNECTION : "{" D+ "," ( "-" | "=" ) ",A1-}"
10 CONNECTION : "{" D+ "," ( "-" | "=" ) ",0}"

```

Table 5.4: BNF-like Antlr grammar for parsing PoLyInfo formula

producing PML. The syntax of this formula is not well documented, but after extensive comparisons between the formula which represents a repeat unit, and the repeat unit for that polymer, the format has been ‘reverse engineered’ and a program was produced which would convert the PoLyInfo formula into PML. This PML can then be converted into CML with the use of JUMBO’s PolymerBuilderTool.

Converting the PML into CML requires the fragments which are used in the PoLyInfo database to be transcribed into CML files. The database provides an enumeration of the fragments with pictures, but no semantic connection tables are available. This means it was necessary to manually produce CML files for these fragments. The most commonly used fragments were transcribed into CML with their attachment point data. To obtain the connection point data unambiguously several examples of the fragment needed to be examined. In some cases there were not enough examples of a fragment to unambiguously allow assignment of the connection point numbering, for example in the case of a large ring which is only attached at one position.

The PoLyInfo formula can be described by Backus-Naur Form (BNF)[62]. The BNF form for PoLyInfo formula using the Antlr[63] parsing program is shown in Table 5.4. In BNF the formal grammar of a language is described in terms of which words or tokens can follow which other words. Line 1 states that a formula consists of the string “plain(” followed by one or more fragmentjoin tokens. Line 2 defines a fragmentjoin as a fragmentid, “/n/” string, a start, an optional end, and one or more side fragmentjoin tokens.

These are all defined further down. The PoLyInfo formula for poly(styrene) is shown below:

```
plain(130/n/{1,-,A1+}{2,-,0}{3,-,0}(708/n/{1,-,0})129/n/{1,-,0}{2,-,A1-})
```

This consists of a formula, which consists of two fragmentjoin tokens. The first has a side chain while the second does not. The first start token is the INITIAL_CONNECTION while the last end token is the FINAL_CONNECTION.

5.2.5 Reaction Processor

The PoLyInfo database contains for each polymer a list of monomers which can be used to synthesise it. In the case that two or more monomers react together to form the polymer they are listed together.

The polymer and monomer files in the PIKS repository were analysed by the `CreateReactionTool`. This created a CML reaction file[18] for each reaction listed in either a polymer or monomer file. This resulted in 11,529 CML reaction files containing 6,150 distinct monomers and 9,357 distinct polymers. These reactions were later used as input to the validation component in Chapter 6.

5.3 Xml2Csv

This project (shown as *XML2CSV* in Figure 5.1) resolves the problem whereby some legacy tools require Comma Separated Values (CSV) or spreadsheet-style input and do not accept XML. This project uses the class `XmlReader` to read in the PIKS XML database for polymer samples, and produce a CSV file as the output. This is achieved through taking a value for each property of a sample, calculating missing values where possible (for example if both M_n and M_w/M_n are known, M_w can be calculated.), and storing the values for each individual sample as a row in a List of HashMaps.

Due to the single-valued nature of a CSV file, only properties which have a single value can be condensed in this way. Properties which vary on more than the composition of the sample, (for example with temperature or time)

cannot be simply converted in this way. Properties have their values converted into SI units if the unit they were stored in was not the SI unit for that property. For example all temperatures are converted to Kelvin from Celsius or Fahrenheit.

The List of HashMaps is then output as a CSV file with one row per sample and one column for each property present in the HashMap. As the dataset is large, each row is written after a sample has been processed. This means that a list of all properties present is required in order to know which properties are missing from each sample so the correct number of columns can be written with a “?” for the missing data. This list of properties is generated after parsing the entire dataset to be used the next time the program is run. This means the program must initially be run twice. If new properties are added they will be added to the list the next time the program runs, and appear in the output when it is run a further time.

The CSV file produced from the PoLyInfo data consists of 40,506 rows and 274 columns. This data file was used in Chapter 7 for machine learning.

5.4 Validation

This comprised the validation component of PIKS. This project (shown as *Compare* in Figure 5.1) compares the monomer data from the PoLyInfo database and the JST database for consistency. It also tests the polymer data, and the reaction data for consistency. Names were extracted for the monomers and polymers using XPath along with their PoLyInfo IDs. These names were converted into SMILES using ChemDraw, and CML using OPSIN. The polymer names were only converted using OPSIN as ChemDraw was unable to parse polymer names.

The structure files generated from the names were used by the Compare program for validation with the PoLyInfo monomer and JST monomer data in PIKS. The validation is run by calling the Compare object on the data. The molecular formulae, molecular weights, SMILES string, and structures derived from the names are then all checked against each other for consistency and the results are outputted as a table of which tests failed for which

monomers and polymers. This component is explained in greater detail in Chapter 6.

5.5 eXist

This comprises the query and database components of PIKS. Once XML data has been produced, for it to be easily searchable it needs to be loaded into an XML database. eXist is a free open-source XML database which was chosen for this purpose. Once the XML data has been loaded into the database, it is then possible to perform XQueries[14] across the entire database. XQuery has some similarities to the Structured Query Language (SQL)[64] used to query conventional databases, but is tailored for XML data structures rather than the traditional table based database structure. An example XML file for a monomer was shown in Table 5.2. An example of the polymer XML was shown in Table 5.1.

The polymer data in the eXist database consists of 69,465 samples over 13,402 polymers. These contain 262,396 properties with 362,325 data elements. The total number of XML elements is 840,279 with 1,338,277 attributes. The indexed nature of the eXist database allows simple XQueries to be computed in a matter of seconds. More complex XQueries on the database such as those in Section A.2 can take over an hour.

The PoLyInfo monomer data consists of 16,371 monomers, with 1,409,393 total elements and 1,408,007 total attributes. The reason the monomer data is so large is due to the MOE descriptors which were calculated and appended to the monomer elements. The reaction data consists of 11,529 reactions comprised of 107,939 total elements and 42,869 total attributes. There are 12,591 JST monomers with 151,840 elements and 125,946 attributes.

5.5.1 XQuery

XQueries are used to retrieve information from the XML database. A common query is simply to ask for all distinct values of a certain property. For example, all the monomer classes. This would be achieved with the XQuery:

```

1 declare namespace cml="http://www.xml-cml.org/schema";
2 <MonomerClasses>
3   {for $c in distinct-values(/cml:molecule/classMembership) return
4     <Class id="{ $c}">
5     { for $m in /cml:molecule[classMembership=/$c] return
6       <Monomer>{string($m/cml:identifier[@convention=
7         "Polyinfo_Monomer_ID"]/@value)}}</Monomer>
8     }
9   </Class>
10 }
11 </MonomerClasses>

```

Table 5.5: Example XQuery which lists monomers by class.

```

1 declare namespace cml="http://www.xml-cml.org/schema";
2 distinct-values(/cml:molecule/classMembership)

```

This XQuery returns the list of monomer classes, which can be seen in Table 4.4. However, much more complicated XQueries can be performed on the data. In order to produce the full darta presented in Table 4.4 the XQuery shown in Table 5.5 was used. This XQuery returns a list of monomer classes, with each class having a sub list of all the monomer IDs which are a member of that class.

This sort of nested query is a useful way of transforming information which was ordered by one variable into a new form sorted by a different variable. Other queries and their results can be found in the Appendix on page 167. These XQueries are performed either using the eXist Java client, or by adding the XQuery as a file in the eXist server, and accessing this file by an http request.

5.6 KNIME

The Konstanz Information Miner (KNIME) [65] is an open source pipelining environment which allows data to be rapidly sent through a workflow and data mining or machine learning tools to be run upon it. This toolkit was used to streamline processes which would otherwise require manual manipulations in a spreadsheet, enabling them to be rapidly repeated on new data as previous components in the data processing pipeline were updated or new data became available. KNIME workflows have been developed to clean

and normalise polymer data, and to generate machine learning models for predicting polymer properties.

Figure 5.2 shows the KNIME workflow which loads the output from the `Xm12Csv` (Section 5.3) program. The CSV file is read into the KNIME node by a `CSVReader`. This KNIME node converts the CSV file into a KNIME table. Each column in a KNIME table is assigned a type; either integer, double or string. This is then cleaned up by selecting only polymers having at least a minimum number of samples, removing descriptors which have too low a variance, normalising various properties, removing samples which have missing values and finally splitting the table into a training set (80%), and a test set (20%).

In Figure 5.3 the training and test sets from the previous workflow are loaded in to memory, and the training set is then split further into a training set (75%) and a validation set (25%). This gives, overall, a 60% training set, 20% validation set and 20% test set. The training set is then used to build a model while the validation set is used to adjust parameters on the model to get the best possible prediction.

Once the parameters have been optimised the test set is loaded and tested against the model. If the test set was used in place of the validation set, then there is the danger that some information from the test set would leak into the parameters of the model and bias the results. The predictions from the model on the test set have their absolute and relative errors calculated, before being saved as a CSV file.

5.7 OPSIN

OPSIN[66] is an open-source tool for converting systematic chemical names to structures, developed by Daniel Lowe. It has been developed for the processing of IUPAC chemical names of general organic molecules into a CML, SMILES or MDL mol file. The initial version of OPSIN could not parse polymer names. The IUPAC source based name does not contain enough information to create the repeat unit structure by itself, specific knowledge of the polymerisation reaction is needed. The IUPAC structure based name,

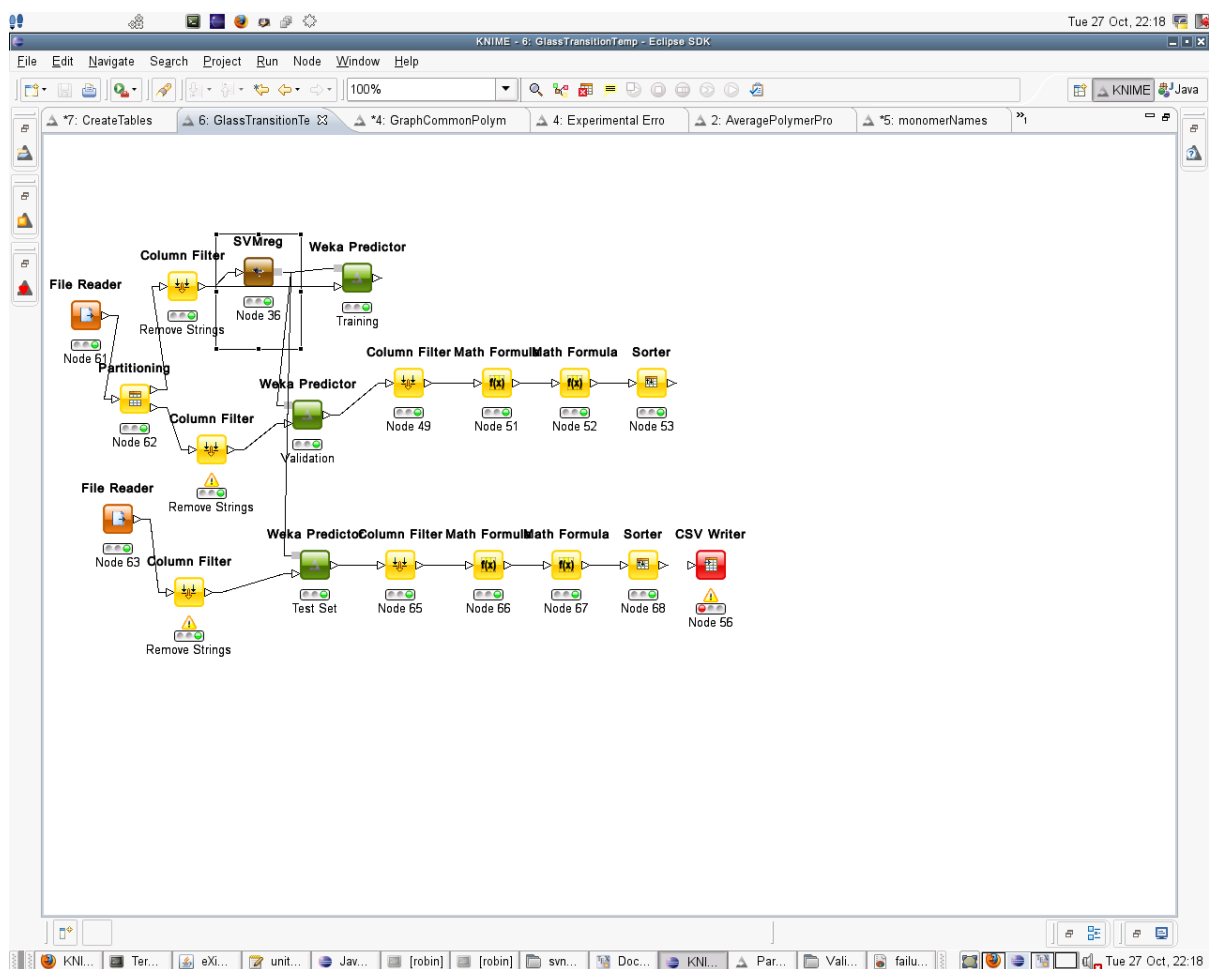


Figure 5.3: This KNIME workflow takes the cleaned up CSV file, and splits it into a training, validation and test set. These are then used to train and validate a model before testing it using the test set.

however, does contain the same information as the repeat unit and so should be amenable to automatic processing to obtain the repeat unit of the polymer.

With assistance from this author, Daniel Lowe has added polymer support to OPSIN, and the latest version (0.9.0) is now capable of parsing IUPAC structure based names. This capability is used in the validation described in Chapter 6, to provide a structure from the name to compare against the structure from the PoLyinfo formula.

5.8 Monomer Substructure Search

The PoLyInfo database has a capability to search the structure of polymer repeat units, but not the monomers. To enable this functionality, a monomer substructure search component was added to PIKS. This takes the form of a webinterface using the Python scripting language. Pybel[67] is used to call OpenBabel[68] for SMARTS[69] and fast sub-structure search capability.

In this implementation a webform allows substructure searches as either a SMILES string for fast sub-structure search or a SMARTS string for SMARTS matching. The query is then run over all the monomers in the database and the resulting monomer SMILES strings and PoLyInfo monomer IDs are returned to the user. The fast sub-structure search uses fingerprints which have been pre-calculated for the monomers to find matches very rapidly, while the SMARTS matching performs a slower search against each monomer in the database, but allows for greater expressibility in search pattern.

5.9 Conclusions

The tools presented in this Chapter allow the extraction of a large quantity of data from both the PoLyInfo database and the JST database. This data is then stored in an XML format which allows complex queries to be run over them. This data can then be investigated (Chapter 4), validated (Chapter 6) and used for modelling (Chapter 7).

Chapter 6

Validation of Chemistry in PoLyInfo database

In this chapter methods for the automated validation of polymer reaction data are discussed. Custom tools were developed for the purpose of analysing the reactions in the PoLyInfo database to investigate the level of error in the system. Inconsistencies in both the monomer and polymer data were investigated before moving on to analysing the reactions to find additional inconsistencies. The aim was to assess the level of error present in the database

6.1 Validation of Monomer data

The monomer data in the PoLyInfo database contains (as described in Section 4.15) a IUPAC name, a SMILES string, molecular weight and molecular formula. It may also have a JST (Japan Science and Technology Agency) [70] or CAS (Chemical Abstracts Service) identification number. In order to investigate errors it is useful to have a secondary source against which to validate data. For this reason the JST numbers were used to retrieve data from the JST database using the tool described in Section 5.2.3.

An XQuery was used to find the total number of monomers with a JST number:

```
1 declare namespace cml="http://www.xml-cml.org/schema";
2 let $m:=/cml:molecule/cml:identifier[@convention="JST_Substance_No."]/@value
```



```
3 return count($m)
```

This gave 12600 monomers. An XQuery to find the number of unique JST numbers replaced line 3 with the following:

```
3 return count(distinct-values($m))
```

which gave 12,532 unique JST numbers. Since this is less than 12,600 there must be some duplicates. In order to find the duplicated identifiers, it was necessary to run a further query:

```
1 declare namespace cml="http://www.xml-cml.org/schema";
2 let $jst:=/cml:molecule/cml:identifier[@convention="JST_Substance_No."]
3 for $id in distinct-values($jst/@value)
4 where count($jst[@value eq $id]) gt 1
5 return $id
```

which gave 61 duplicated JST numbers show in Table 6.1. In the vast majority of these cases the JST number resolves to a molecule which matches one of the two PoLyInfo monomers corresponding to that JST number. The other molecule has a different structure. These JST number assignments in the PoLyInfo database are therefore incorrect.

Finding duplicates will only reveal the incorrect assignment of JST numbers when there is already a correct assignment to the JST number. In order to find all the errors it is necessary to do a more in depth analysis. The JST data was parsed for a molecular weight, chemical formula and, in some cases, a structure file in mol format. This was then used for validation of the monomer data against the PoLyInfo data as well as internal validation for the JST database.

An additional source of data for the validation procedure is the chemical name of the monomer. The monomers in the PoLyInfo database all have an IUPAC name (although not necessarily the correct name). Using OPSIN[66] (developed by Daniel Lowe) and ChemDraw[71] structures were generated from the IUPAC name of the monomer. These structures were then compared to the structure given by the SMILES string from the PoLyInfo database, as well as the chemical formula and molecular weight from PoLyInfo and JST. This process is shown in Figure 6.1.

J1.513.354F	J114.860E	J907.565H
J761.003C	J799.778G	J124.160E
J1.514.017H	J4.202A	J3.530K
J801.588K	J	J1.336.148G
J799.787F	J208.167I	J1.348.567D
J299.149G	J99.304B	J5.088A
J208.953J	J1.517.595H	J801.359D
J45.018I	J802.243G	J799.955K
J4.298F	J1.341.456D	J53.760H
J213.631G	J1.341.620F	J36.791E
J195.318D	J227.378K	JJ1.918.024G
J6.182D	J1.517.044A	J6.042I
J799.907K	J36.771K	J802.012D
J801.248B	J208.715D	J798.571A
J40.865D	J1.513.415A	J1.336.146K
J10.947I	J141.640E	J6.183B
J799.895C	J273.970D	J802.809E
J803.074J	J658.490J	J23.095B
J1.341.931K	J9.352A	J993.945H
J802.723D	J6.196D	
J80.318I	J1.341.127A	

Table 6.1: Table of the duplicated JST numbers in the PoLyInfo database.

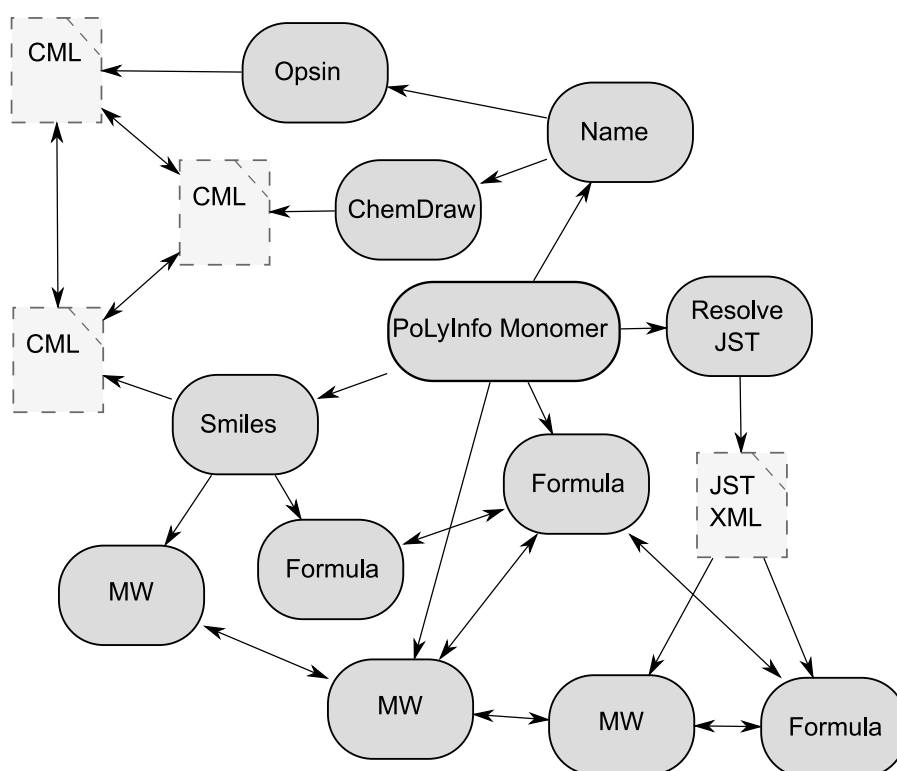


Figure 6.1: The process of validation of the monomer data.

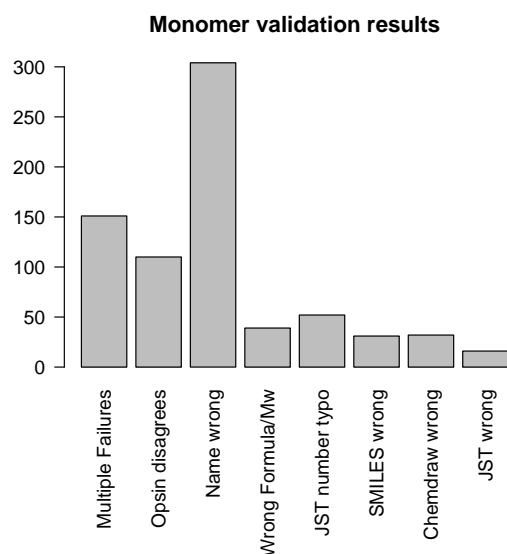


Figure 6.2: Types of monomer failure. If all the tests passed except from one source of data, that source was identified as the cause of failure. Multiple failures where the cause could not be identified were classified into Multiple Failures.

6.1.1 Incorrect Monomer Data

Out of the 12,591 monomers with a JST reference number, 735 (5.8%) have at least one inconsistency in their data. These are due to a variety of reasons. These include the formula being unparsable (or just outright wrong), and the name not matching the structure (either because the name is ambiguous or invalid, or because the SMILES string is wrong.)

A further source of errors is the fact that all the structures use the implicit hydrogen convention. This is where the structure is given without the hydrogen atoms explicitly stated. While for most hydrocarbons the number of hydrogens attached can be easily determined from the bonding, for some complex ring systems with nitrogens different programs produce different answers when parsing the structure. This can lead to a different number of hydrogens in the final molecule compared to the molecular formula. If chemical structures always stored the hydrogens explicitly, this problem could be avoided.

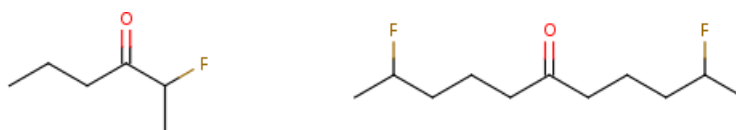


Figure 6.3: This shows the importance of a space in a chemical name. The structure on the left is “1-fluoroethyl propyl ketone” while the structure on the right is “1-fluoroethyl(propyl) ketone”. OPSIN produced the structure on the right for “1-fluoroethyl(propyl) ketone” while ChemDraw produces the structure on the left.

The results were classified based on which tests failed and which passed. The frequency of each test failure pattern was recorded and classified if the cause of failure could be determined. For example if all the tests passed except for comparisons with the molecular weight, then the molecular weight was regarded as being incorrect. In Figure 6.2 the different causes of error are shown. The 151 multiple failures were when many different comparisons failed so it was impossible to assign any one cause.

OPSIN Disagrees

The 110 OPSIN disagrees cases were when all tests passed, other than the structure produced by OPSIN. This not necessarily mean that OPSIN parsed the name incorrectly, as in some of these cases the name was wrong, with ChemDraw parsing the name incorrectly to produce the structure in the PoLyInfo Database. This frequently happened with ketone names. A name such as “1-fluoroethyl(propyl) ketone” will be parsed by Chemdraw as “1-fluoroethyl propyl ketone” resulting in a 6 carbon molecule, while OPSIN will produce an 11 carbon molecule. This is shown in Figure 6.3. This name is ambiguous, and it would be better if the space was included to remove the ambiguity.

Other cases were due to bugs in OPSIN which have been reported to Daniel Lowe and will be fixed in the next version of OPSIN.

Wrong Name

304 of the monomers tested failed as their name was wrong. This classification happened when both ChemDraw and OPSIN interpreted the name differently to the PoLyInfo database SMILES, but the SMILES agreed with the other data. For example monomer M0801073 has IUPAC name “1-propenyl(pentafluoropropyl) ketone” (Figure 6.4). As in the previous section, this was interpreted by OPSIN with the 1-propenyl referring to the pentafluoropropyl. This resulted in a doubling around the ketone. ChemDraw interprets the name in the same manor as “propenyl pentafluoropropyl ketone” where the space is important to specify that the two ligands are attached on either side of the ketone rather than the propenyl applying to the pentafluoropropyl. The SMILES from the PoLyInfo database,

```
CC=CC(=O)C(F)(F)C(F)(F)C(F)(F)F
```

consists of a ketone where one side has a heptafluoropropyl group and the other side a propenyl group. In this case the PoLyInfo database has named the molecule with “pentafluoro” when it meant “heptafluoro”, as well as the previous missing space in the name.

The molecule is linked to the JST database entry, J799.494J, which has 7 fluorine atoms and matches the SMILES string given in the PoLyInfo database. Since the SMILES matches the JST reference, it is likely that the name is incorrect.

Formulae

The 39 samples which are classified as failing Formula/Mw have inconsistent molecular weight or formula. There is no convention for how to represent isotopes in a molecular formula. The Hill System[72] is universally used for molecular formulae but pre-dates isotopic enrichment and so no provision is made for representing isotopes of samples.

The PoLyInfo database and the JST database have chosen to use different conventions for representing isotopes in their formulae, which is producing these errors. A ethene with one C^{13} would be represented in the JST database by C1H4[13C] while the PoLyInfo database would represent it as C2H4[13C].

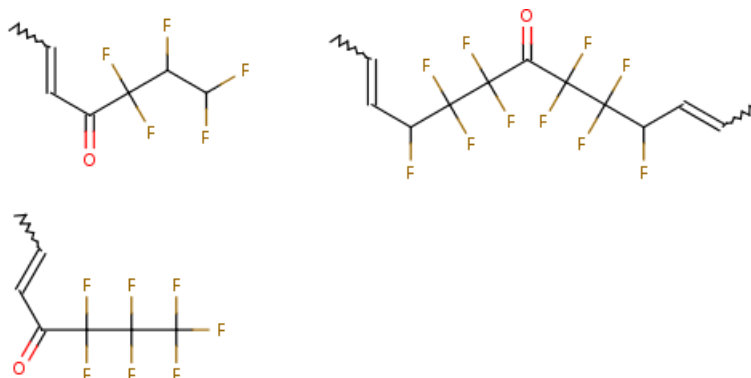


Figure 6.4: The molecule on the top left is produced by ChemDraw, the molecule on the top right by OPSIN and the molecule on the bottom is the SMILES string given in the PoLyInfo database for “1-propenyl(pentafluoropropyl) ketone”. Clearly the name does not match the SMILES as the SMILES contains 7 fluorine atoms while the name has only 5. The name is also ambiguous and can be interpreted in two different ways, giving the OPSIN and ChemDraw structures. If there was a space after the “propenyl” the name would not be ambiguous.

JST errors

There were 52 JST typo errors, and 16 wrong JST errors. The JST typo errors occurred when the JST number listed in the PoLyInfo database either had a double “J” at the start or an old JST number. When the old JST number was looked up in the JST database the file returned identified itself with the new JST number. It appears that the JST numbers have been updated at some point, but at the moment the old numbers can still be used to retrieve the correct molecule. It is unknown for how long the old numbers will continue to be supported.

The 16 wrong JST errors were due to the molecule returned from the JST database not matching the data from the PoLyinfo database, or the structures produced by OPSIN and ChemDraw. Out of the 61 duplicated JST numbers previous mentioned in Table 6.1 for 16 of these this was the only mistake, while the remainder had multiple errors and so were classified into the multiple errors class.

SMILES

There were 31 cases solely due to the SMILES being wrong. This was either due to the SMILES being invalid, for example a common error was a trailing = at the end of the string. The monomer M1532203 (2-(4-methoxyphenyl)-1,3-oxazoline) has SMILES string:

```
COc1ccc(cc1)C1=NCCO1=
```

This SMILES is correct for the monomer if the trailing = sign is removed. Since a = indicates a double bond, a = at the end of a SMILES string is meaningless. How this error was introduced into the database is unknown.

Another error is the replacement of the [Si] string for a Si. This means that instead of the element silicon, a sulphur attached to an aromatic iodine is used instead. This makes no chemical sense since an aromatic iodine does not exist. An example of this is the monomer M0531257 (3-(trimethylsilyl)ethynylstyrene) which has SMILES,

```
C=Cc1cc(ccc1)C#CSi(C)(C)C
```

This error presumably could only have occurred if the SMILES was typed in manually, since the use of a drawing package would make the aromatic iodine selection impossible, as well as a two atom substitution for one atom would be an unlikely miss-click. In the case of M2330878 the SMILES string has an upper-case I:

```
CSI(C)(C)NCCCCNSI(C)(C)C
```

Again, the SI should be replaced with a [Si] to make the monomer structure agree with the name.

ChemDraw wrong

Out of the 32 instances of ChemDraw getting the wrong structure, in 17 cases the wrong isomer was produced. The remaining 15 cases had the wrong molecular formula.

In the cases where ChemDraw obtained the wrong isomer, a large proportion were from compounds containing “bisphenyl” or “bispyridyl”. For example the monomer M2871578, “biphenyl-3,3’-diylbis(carbonyl-p-phenylene)

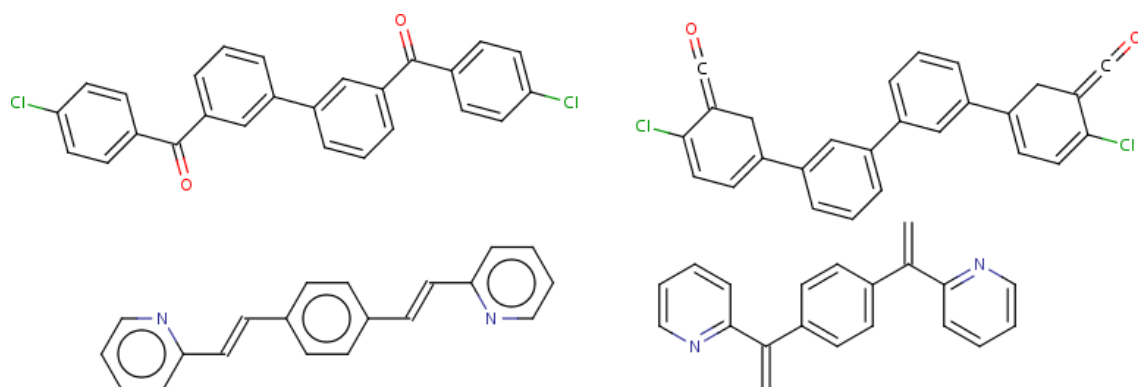


Figure 6.5: The molecules on the left are produced by OPSIN, the molecules on the right by ChemDraw. The top pair of molecules are for “biphenyl-3,3’-diylbis(carbonyl-p-phenylene) dichloride” and the bottom pair are for “1,4-bis(2-pyridylvinyl)benzene”. ChemDraw produces the wrong isomers in these cases.

dichloride”, is shown in Figure 6.5. The molecule on the left is how OPSIN and the PoLyInfo database represent the molecule, and on the right is the representation from ChemDraw. ChemDraw has turned the bridging carbonyl groups into bizarre additions to the ring. Another example is the monomer M0502189, “1,4-bis(2-pyridylvinyl)benzene”, also shown in Figure 6.5.

Out of the 15 cases where ChemDraw produces the wrong molecular formula, some are a strange reversal of the ketone case earlier. Monomer M0502216, “ethylstyryl ketone” is shown in Figure 6.6. Here OPSIN has interpreted the name as “ethyl styryl ketone” while ChemDraw has produced “(ethylstyryl) ketone”. This further shows that great care must be taken to avoid creating ambiguous names which can be interpreted in two different ways. The reason for the reversal in both OPSIN and ChemDraw’s behaviour is unclear.

OPSIN produces multiple parses of a chemical name and chooses the most likely, so the lack of a locant on the styryl means it will be unsure where to attach the ethyl group (if it were attached to the styryl) hence it attaches it to the ketone instead. Since OPSIN parses “propanol” as “propan-1-ol” the lack of locant in the previous examples might have defaulted to 1, while there

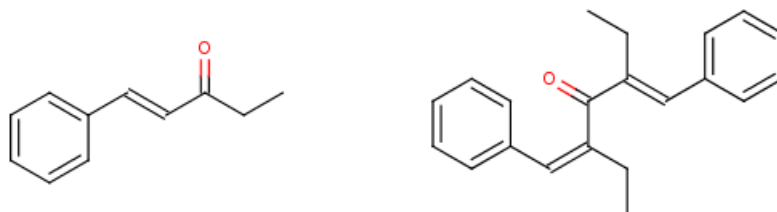


Figure 6.6: The molecule on the left is from the PoLyInfo database and OPSIN, the molecule on the right by ChemDraw for “ethylstyryl ketone”.

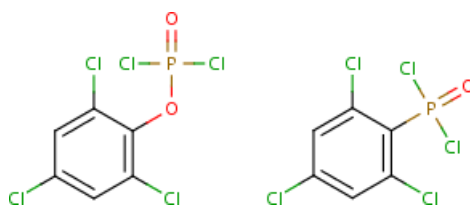


Figure 6.7: The molecule on the left is from the PoLyInfo database and OPSIN, the molecule on the right by ChemDraw for “2,4,6-trichlorophenyl dichlorophosphate”.

might be no default locant for a styryl group.

The other ChemDraw errors involve phosphinate groups. In M3030139, “2,4,6-trichlorophenyl dichlorophosphate” ChemDraw loses an oxygen from the structure. This is shown in Figure 6.7.

6.2 Validation of Polymer Data

The polymers’ chemical structures are represented by repeat units. These repeat units have been parsed from the PoLyInfo formula into CML as described in Section 5.2.4. In order to validate these they need to be compared against another representation of the repeat unit. The polymers also have an IUPAC structure-based name. This name can be parsed into a CML file using OPSIN. The repeat unit from the PoLyInfo formula can then be compared against the repeat unit generated from the IUPAC structure based name.

Repeat units cannot be simply compared with the same method as a small molecule, due to the many ways a repeat unit can be represented

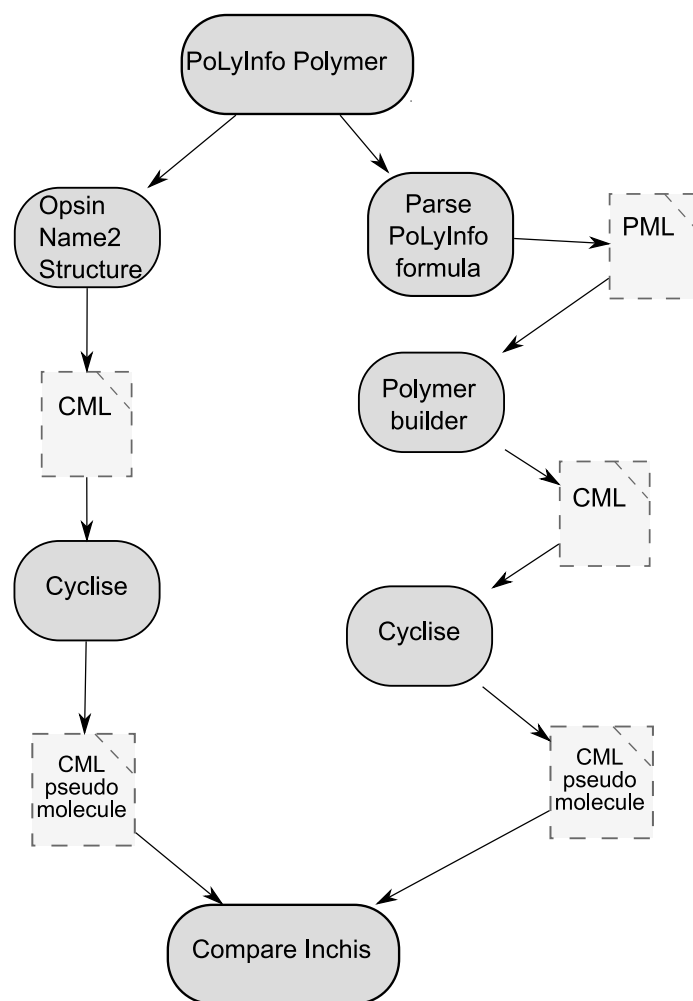


Figure 6.8: The process of validation of the polymer data.

depending on where the R groups are placed. A polymer which has a repeat unit of R-A-B-C-R should match a repeat unit of R-B-C-A-R, which a naive comparison will fail to do. In order to facilitate the comparisons of repeat units a method of pseudo-cyclisation was adopted. The atoms which are bonded to R groups had a new bond created between them. This then allows traditional connection table comparison methods to work on the repeat units. To compare the cyclised repeat units, InChIs[73] were generated for both molecules which were then compared. As the chirality was not always known the comparison was done disregarding the chiral layer. This process is shown in Figure 6.8. Out of the 6,508 polymers for which repeat units were generated by both methods, 6,063 had matching InChIs. The remaining 443 did not match. This gives 7.3% error in the polymer structural data.

6.2.1 Incorrect Polymer Data

Out of the 443 which did not match, there were a variety of errors. Typically the name simply did not match the polymer repeat unit as it was lacking in a few important atoms. For example P030089 is shown in Figure 6.9. The IUPAC name given in the database is “poly[1,1,2-trifluoro-2-(trifluoromethyl)ethylene]” which results in the structure in the middle. The PoLyInfo formula is:

```
plain(188/n/{1,-,A1+}{2,-,0}{3,-,0}(106/n/{1,-,0}{2,-,0} 191/n/{1,-,0})
      192/n/{1,-,0}{2,-,A1-})
```

This results in the structure on the left. The intended structure is the one on the right as the monomer listed for this polymer is shown on the right. Since the monomer would polymerise to the repeat unit on the left the structure on the left (from the PoLyInfo formula) is probably correct while the name is wrong.

Other type of disagreement between name and structure was the attachment points of substituents on aromatic systems. The two structures for P100818, “poly[imino-5-(phenylsulfonyl)isophthaloylimino-1,4-phenyleneoxy-1,4-phenylene]”, are shown in Figure 6.10. The one on the left is from the IUPAC name via OPSIN, while the one in the right is from the PoLyInfo

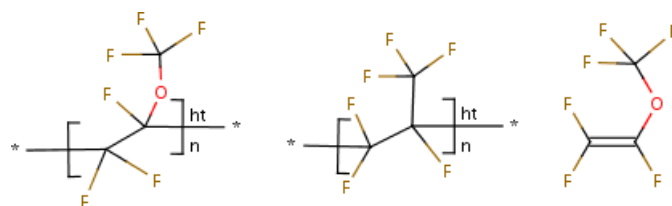


Figure 6.9: The two structures for P030089. The one of the left is from the PoLyInfo formula. The one in the middle is from the IUPAC structure-based name via OPSIN. On the right the monomer given for this polymer is shown indicating that the structure on the left is the correct one and that the name is wrong.

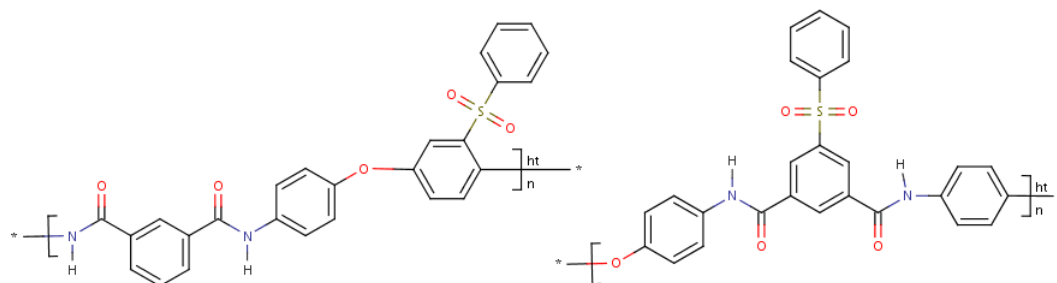


Figure 6.10: The two structures for P100818. The one of the left is from the IUPAC name via OPSIN, the one on the right is from the PoLyInfo formula. The name is slightly ambiguous, but the correct structure is on the right.

formula. The only difference is the position of the phenylsulfonyl group.

Here the name is slightly ambiguous, and could be written more explicitly as “poly[imino-(5-(phenylsulfonyl)isophthaloyl)imino-1,4-phenyleneoxy-1,4-phenylene]” to give the correct structure. Without this extra bracket OPSIN is assuming the 5 refers to the right-most group (as it normally does in small molecules), while in this case it refers to the left-most group. Improved heuristics for parsing ambiguous polymer names in OPSIN could remove this error in the future.

6.3 Validation of the Chemistry

The PoLyInfo database contains 14,008 different chemical reactions¹. In each reaction one or more monomers react to form a polymer. Each monomer can belong to one or more of 38 different monomer classes. The reactions themselves can belong to one of 6 classes and the polymers can belong to one or more of 22 polymer classes. These different classes are shown in Table 6.2.

The reactions classes are defined by the PoLyInfo database as follows:

Addition polymerization Only the unsaturated bond reacts, and there is no elimination component.

Ring-opening polymerization A polymerization with an ring-opening of the ring monomers without any elimination component.

Polycondensation A polymerization with an elimination of a component.

Polyaddition A polymerization without elimination components other than the above-mentioned.

Polymer reaction A chemical reaction in which at least one of the reactants is a high-molar-mass substance.

These reactions were converted into XML to create a searchable database.

As well as validating the monomers and repeat units individually, it is also possible to validate the monomers against the repeat units. For some polymers such as polyolefins this is relatively straightforward as there are the same number of atoms in the monomer as the repeat unit. Some care must be taken for polymers where the repeat unit is half the size of the monomer, such as polyethene. In these cases it is necessary to compare the monomer against a dimer of two repeat units joined together. Two examples where this is the case can be seen in Figure 6.11. In the top example, the ethene monomer contains two carbon atoms, but the polymer formed, poly(ethene), has a repeat unit with only a single carbon atom. In the lower example each

¹This does not include random copolymers which were excluded from this study.

Monomer Classes	Polymer Classes
Cyclic olefines Other monomers Diamines Sulfur contg compds Cyclic imides Silicon contg cyclic compds Dicarboxylic acids Diols Melamines & Ureas Cyclic amines Dihalides Amino acids Cyclic acid anhydrides Silane compds Cyclic sulfides Dienes Vinyl compds Acrylic acids Phosphorus contg compds Acetylenes Olefins Cyclic ethers Aldehydes Anilines Diketone Cyclic iminoethers Phenols Phosphorus contg cyclic compds Carbonates Styrenes Arom ethers Halo-olefins Lactones Diisocyanates Hydroxy acids N-carboxy anhydrides Lactams Cyclic carbonates	Other polymers Polyphenylenes Polyimines Polysulfones/sulfoxides/sulfonates/sulfoamides Polyketones/thioketones Polyimides/thioimides Polysiloxanes/silanes Polyamides/thioamides Polyoxides/ethers/acetals Polyurethanes/thiourethanes Polyesters/thioesters Polysulfides Polyanhydrides/thioanhydrides Polyhalo-olefins Polyvinyls Polyacrylics Polyolefins Polydienes Polyphosphazenes Polycarbonates/thiocarbonates Polystyrenes Polyureas/thioureas
	Reaction Classes
	Ring opening polymerization Polycondensation Addition polymerization Polyaddition Polymer reaction Junction Unit

Table 6.2: Table of the different classes defined in the PoLyInfo database.

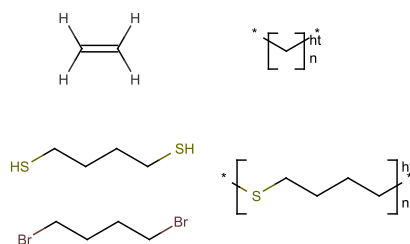


Figure 6.11: Examples where a repeat unit is half the size of the monomer. The top repeat unit is poly(methylene) (P010001). The lower repeat unit polymer is poly(sulfanediylbutane-1,4-diyl) (P080018). The monomers shown produce two units of repeat unit when polymerised.

monomer has four carbon atoms. The polymer, poly(sulfanediylbutane-1,4-diyl), only has four carbon atoms in the repeat unit rather than the eight found in the monomers.

6.3.1 Data selection

In order to facilitate accurate information extraction from analysis of the chemistry, some data selection was used. Only the polymer reactions where both the monomer and the repeat unit were judged to be self-consistent from the previous validation were selected for analysis. This was to ensure that any errors in the polymer reactions were not due to underlying errors in the structure of the repeat units or monomers. This process can be seen in Figure 6.12. The resulting set of 4,754 reactions will be used as the set of reactions under test for the rest of Section 6.3. These reactions consist of 2,366 distinct monomers 3,368 distinct polymers. Strangely out of the 16,371 total monomers in the PoLyInfo database, 7,630 are not involved in any reaction. These reactions were saved as CML reaction files, producing a corpus of 4,754 polymer reactions in CML.

Some of these reactions are labelled as being of type “polymer reaction” by the PoLyInfo database. This is defined to mean: “A chemical reaction in which at least one of the reactants is a high-molar-mass substance” by their help pages. In these polymerisation reactions some other reaction has also occurred, for example altering side-chains. This means that the reaction

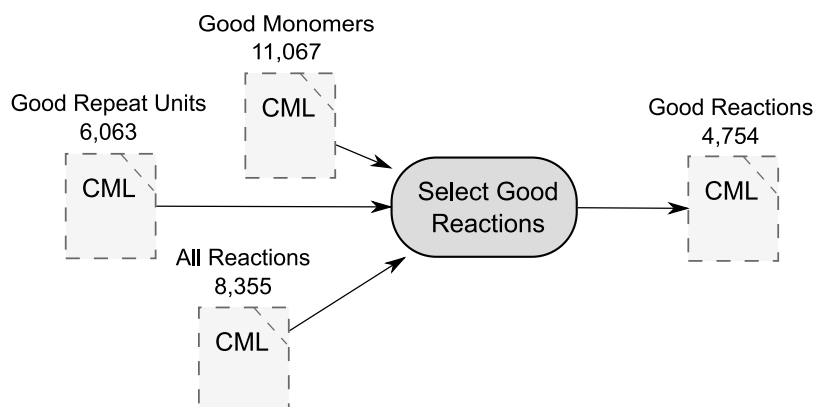


Figure 6.12: The process used to select ‘good’ reactions. Only reactions which both had exclusively ‘good’ monomers and a ‘good’ repeat unit were selected as a ‘good’ reaction. The monomers and repeat units were defined to be ‘good’ if they contained no inconsistent data.

describes more than the monomers polymerising to form a repeat unit and so they should be analysed separately.

6.3.2 Reactions

The polymer reactions were classified using a Java program. The reaction was read in as a CML reaction file which was then classified according to the molecular formulae of the repeat unit and monomers. The molecular formulae difference between the monomers and the repeat unit was calculated using a Java program based on the JUMBO toolkit. If the repeat unit contained more atoms than were present in the combined monomers then the reaction was classified as ‘bad’. For example if there was an element present in the repeat unit which was not present in the monomers. If the molecular formula of the repeat unit and the monomers matched exactly, apart from the two R groups in the repeat unit, the reaction was classified as ‘noDifference’; for example most olefin or ring-opening polymerisations. If there was a loss of atoms from the monomers to the repeat unit, the reaction was classified as ‘lossOfSmallMolecule’; for example nylon 6 polymerisation loses an H_2O . This is shown in Figure 6.13.

The more complicated case was where the repeat unit could be doubled

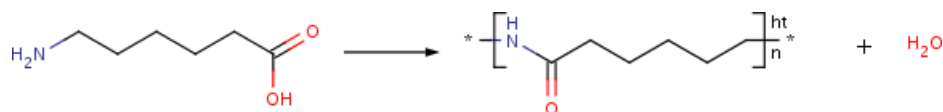


Figure 6.13: Nylon 6 monomer, repeat unit and small molecule produced in the reaction.

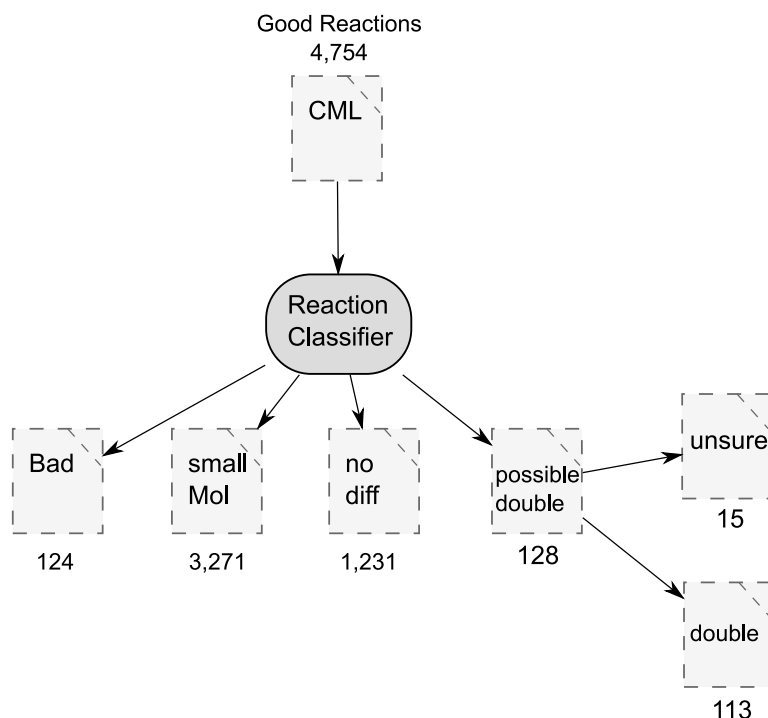


Figure 6.14: Diagram to show the classification of the reaction data.

without causing any atoms to be gained between the monomers and repeat unit. In this case the reaction was classified as a ‘possibleDouble’. In order to see if the reaction should be double, the remaining molecular formula difference was compared with the molecular formula differences from the ‘lossOfSmallMolecule’ reactions. If the doubled formula yielded either no difference, or a difference which matched one from the ‘lossOfSmallMolecule’ reactions, which are assumed to be valid leaving groups, then it was regarded as a “double”, while if it did not match then the reaction was regarded as an “unsureDouble”. This is process shown in Figure 6.14.

An example of a suspect double is shown in Figure 6.15. In this case the

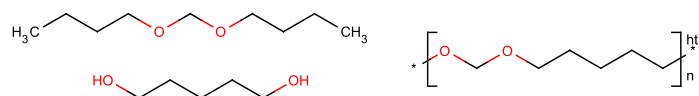


Figure 6.15: An example of a possible double which is not a double is poly(methyleneoxypentamethylene oxide) (P070091). In this case the monomers are losing a large enough number of atoms that a second repeat unit could be made, but then the resulting mass loss does not represent a reasonable leaving group (C_2H_8). Since C_2H_8 is not found in the list of ‘lossOfSmallMolecule’ classified reactions this reaction is not a double.

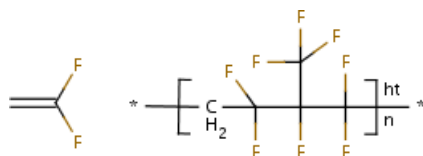


Figure 6.16: An example of a bad reaction. The repeat unit contains more atoms than the monomer.

reaction is not a double, it simply has very large leaving groups. When the repeat unit is doubled and compared to the monomers two carbon atoms and eight hydrogen atoms are left-over. Since the list of small molecules lost from the ‘lossOfSmallMolecule’ classified reactions does not contain C_2H_8 the reaction is classified as a suspect double. In this case the low atom efficiency of the reaction caused the monomers to be larger than two repeat units.

6.3.3 Results

The results are summarised in Table 6.3. This gives, if we count the unsure-Double as possible errors, the percentage error in the reactions as 2.9%. This is much lower than the 15% error in the polymer structure data.

An example of the bad reactions is shown in Figure 6.16. This polymer repeat unit only has one sample which is from a paper from 1979 by Schoenbacher et al. [74]. There appears to be no mechanism which would produce the polymer repeat unit from the given monomer, since the repeat unit contains far more atoms than the monomer. As this polymer is only reported once in 1979 and not since, it is very likely that the structure in the database for the repeat unit is wrong.

Type	Count
noDifference	1231
lossOfSmallMolecule	3271
double	113
unsureDouble	15
bad	124

Table 6.3: Table of the reactions in PoLyInfo database

One of the common small molecule leaving groups is H_2 . In these cases the actual leaving group was H_2O , normally from a reaction involving aromatic rings. These reactions are normally specified as being conducted under an oxygen atmosphere, but this is not show in the PoLyInfo database as a co-monomer.

6.4 Conclusions

The structural data in the PoLyInfo database contains considerable errors. The monomers suffer from a 5.8% error rate, the polymers from a 7.3% error rate and the reactions a 2.9% error rate. This is excluding the reactions between monomers and polymers which were known to contain errors. The error rate in the reactions would be higher if it was calculated over the entirety of the data.

The presence of these errors suggests that the database does not have an automated validation system for checking the integrity of the data it contains. If such a system was implemented then these errors could be fixed. The monomer SMILES contain a mixture of errors which suggests they are manually typed in. Since silicon is represented as both “[Si]”, “Si” and “SI” it is unlikely to be a computer program causing the error. Since the molecular formula does not always agree with the SMILES it appears that the monomer structural data is being held in two separate ways which is how these errors arise.

The reactions and monomers in the PoLyInfo database do not have any provenance, unlike the polymer samples which all have a reference to the

literature. This means that if a bad reaction is added to the database there is no way to tell where this reaction came from if there are multiple samples of that polymer. Due to the powerlaw-like distribution of samples of polymers, however, in a large number of cases there is only one sample of a polymer and so the reaction can be assumed to have come from the same reference which allows nonsensical reactions to be checked against the literature. In the case of there being many samples of a polymer there is no way of finding the paper which might contain some exotic catalyst or unusual reaction conditions. More importantly if a paper were to be retracted, while it would be easy to remove the polymer from the database, removing reactions which had been added solely due to that paper would be an impossible task.

Chapter 7

Machine Learning Analysis of PoLyInfo Data

7.1 Aim

In this Chapter some of the obstacles facing polymer property prediction are discussed. Data taken from the PoLyInfo database into Polymer Informatics Knowledge System (PIKS) are processed and used to create a dataset from which models for the prediction of polymer properties can be made. The data was then split into training, test and validation sets. Support Vector Regression (SVR) models were produced to model the glass transition temperature of a sample of a polymer. This was done both with and without sample characterisation data to see how well the variation between samples could be modelled.

7.2 Variation

Polymer properties have been predicted using a variety of methods[75]. A large number of the approaches taken have had one assumption in common, which is to assume that the properties of a polymer depend only on the type of polymer rather than on any sample properties [76, 77, 78, 79] such as molecular weight, the molecular weight distribution, degree of branching,

PoLyInfo ID	Name	Samples
P010001	Polyethene	2879
P010002	Poly(prop-1-ene)	1768
P020001	Polystyrene	2268
P040048	Poly(methyl methacrylate)	1499
P070013	Poly(ethylene oxide)	1314
P090027	Poly(ethylene terephthalate)	1207
P460064	Polyaniline	1594

Table 7.1: The most common polymers in the PoLyInfo database by sample count.

crystallinity or tacticity. No matter how well a method can predict the properties of a type of polymer, to be able to predict the properties of a particular sample of that polymer will always have an error which depends on the spread of values of a property between samples of that polymer.

Some property prediction methods do take into account the molecular weight of the polymer when predicting properties such as intrinsic viscosity which does vary strongly with molecular weight [80]. This allows different values to be predicted for different samples of the same polymer. Without predictions depending on the sample properties the best a model can do is predict the mean value for that polymer. How well such a model scores depends on the scoring metric and the standard deviation of the property across the samples.

The variation in the values of a property, such as the glass transition temperature, varies both within a polymer and between different types of polymers. When all the data in the PoLyInfo database is aggregated it is clear that there is a lot of variation between samples of the same polymer type. In Figure 7.1 the glass transition temperatures of the most common polymers in the PoLyInfo database have been plotted as a boxplot. The source-based names for these polymers are shown in Table 7.1. From the boxplots it is clear that the variation between samples of the same polymer is very important when trying to predict the glass transition temperature of a polymer sample.

In Figure 7.2 the variation in glass transition temperature across all the

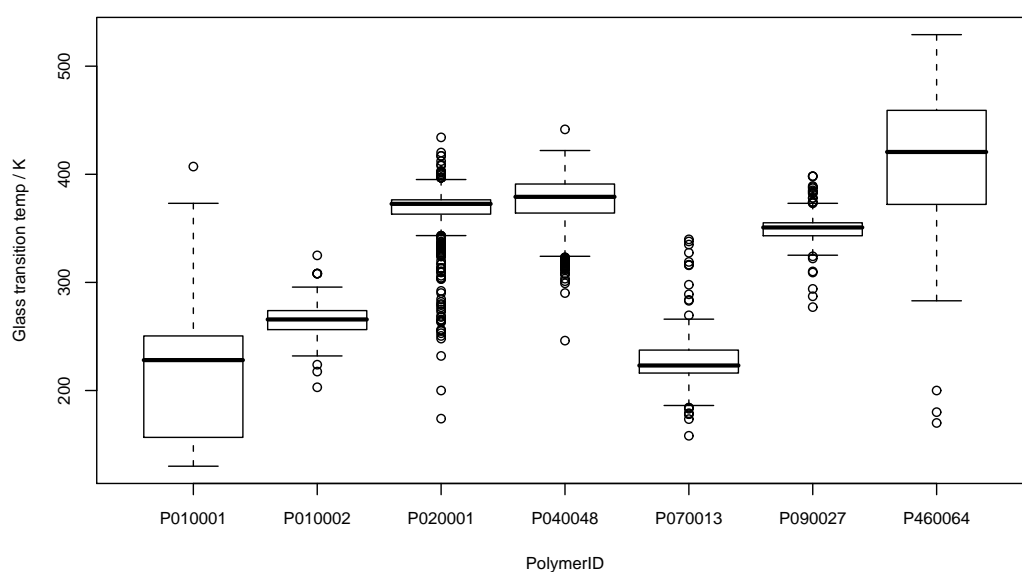


Figure 7.1: The variation in glass transition temperature between the different most frequently occurring polymers in the database. The whiskers are drawn to the highest or lowest datum within a distance of $1.5 \times$ Inter-Quartile Range of the 3rd or 1st quartile respectively. Data outside this range is shown as an outlier. Table 7.1 shows the IUPAC source based name for these polymers.

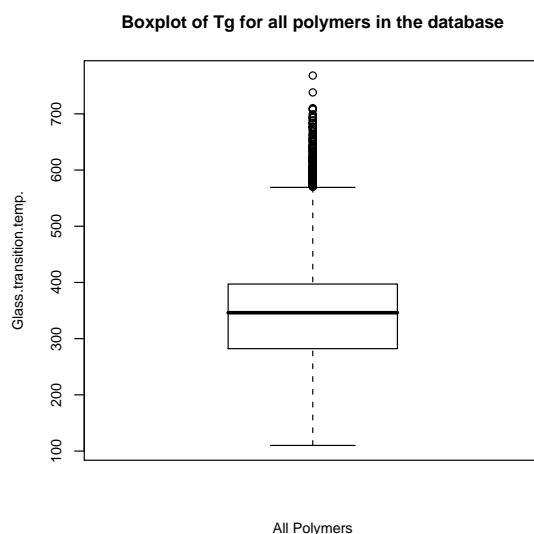


Figure 7.2: The variation in glass transition temperature between all the points in the database.

polymers in the database is shown. (Figure 7.3 shows the same data as a histogram.) From these graphs we can see that the overall spread of polymer data is much larger than that of individual polymers. This shows that the type of polymer is still important for predicting properties. However, there is a wide sample variation within a polymer type.

In order to generate different predictions for different samples of the same polymer models need to take into account sample properties of the polymers. The characterisation data which are recorded includes average molecular weight, polydispersity, crystallinity, branching and tacticity. Unfortunately the proportion of samples which include full characterisation data is quite small, but a large number of samples have both a molecular weight average and a polydispersity index such as M_w and M_w/M_n while a smaller number record the crystallinity.

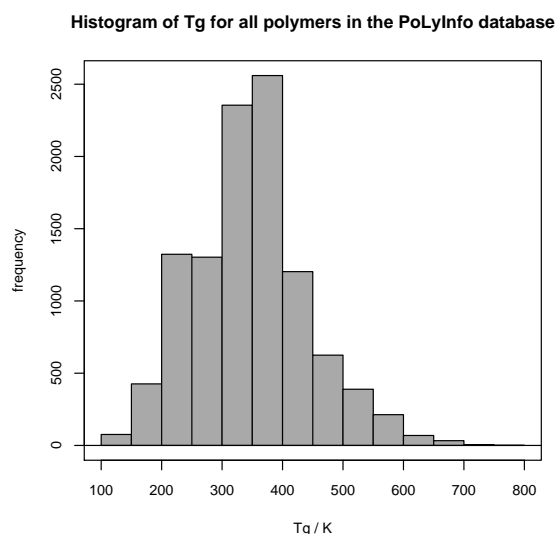


Figure 7.3: The variation in glass transition temperature between all the points in the database shown as a histogram.

7.3 Cause of Variation

An important question is whether the variation in sample properties of a given polymer is intrinsic to the type of polymer, or whether it is an artefact of human experiments. If a given polymer exhibits a large range of property values, is this wide range due to the amount of human ingenuity which has been used to customise its properties, experimental error, or is it some intrinsic variability of the polymer? In order to investigate this, time-dependant property data was plotted. In Figure 7.4 the density of samples of poly(ethene) over time are plotted. The majority of samples stay within a wide band, with a few new samples possessing drastically different values. These very low density samples have been created using a nitrogen-gas injection technique by Zotefoams Plc to create very low density foam[81].

In Figure 7.5 the density of samples of poly(methyl methacrylate) over time are plotted. In this case there is much less variation over time compared to the poly(ethene) plot in Figure 7.4. However this does not mean that a low-density foam cannot be created, simply that it has not been reported in the literature which has been added to the PoLyInfo database. We must

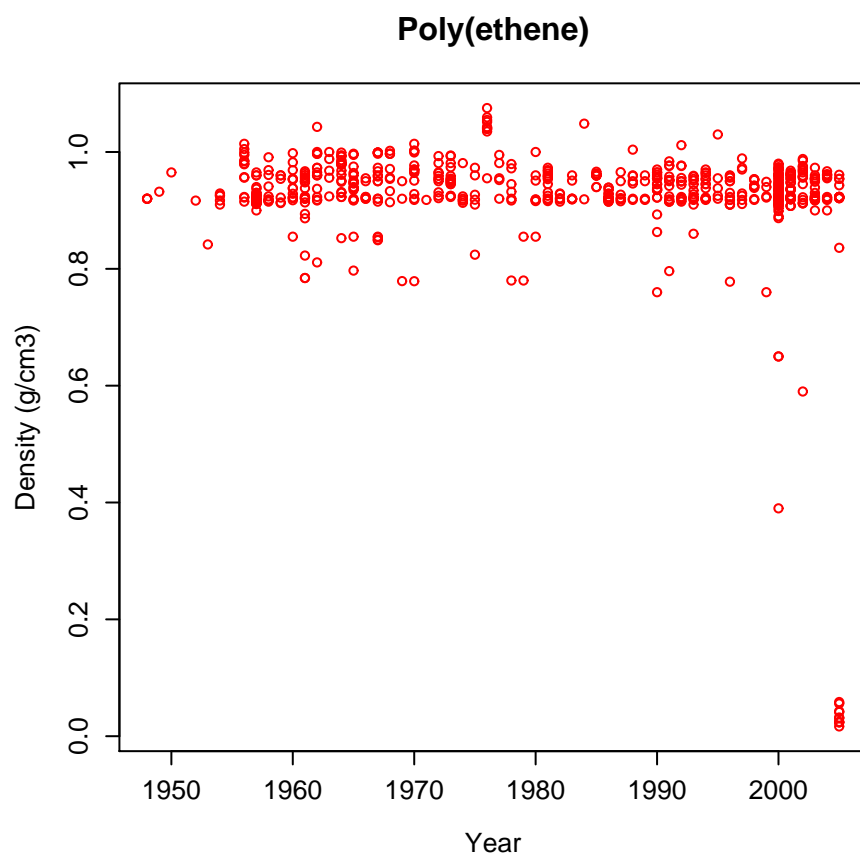


Figure 7.4: Density of samples of poly(ethene) plotted against year of publication. As new techniques are developed a few new, lower density samples have been created.

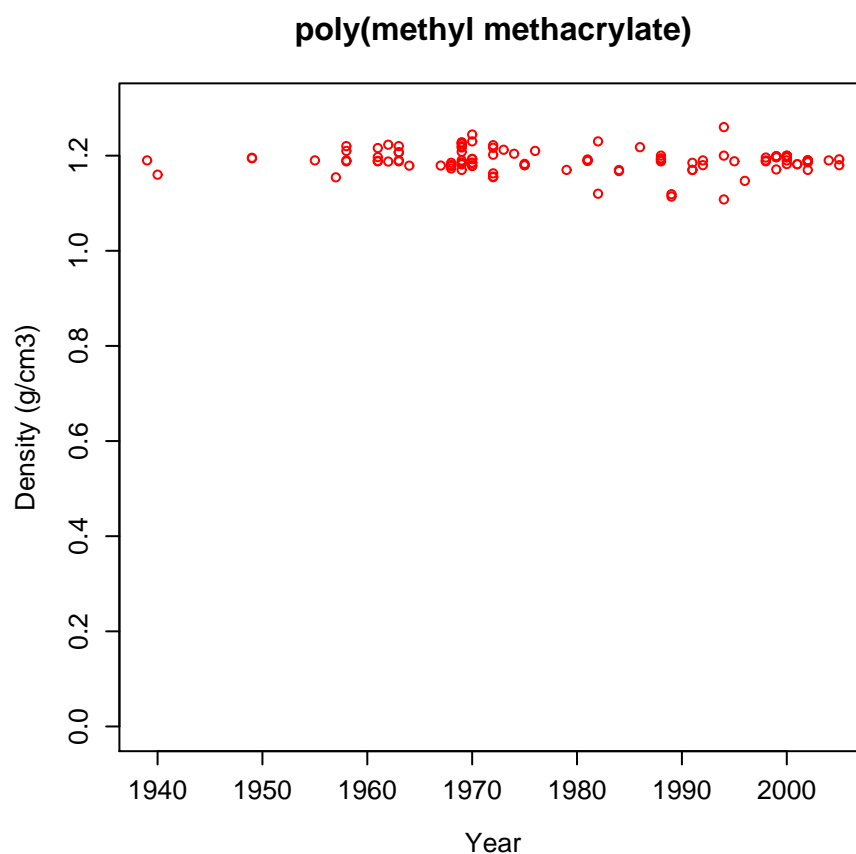


Figure 7.5: Density of samples of poly(methyl methacrylate) plotted against year of publication. This polymer exhibits a smaller degree of variation over time than for poly(ethene) in Figure 7.4.

therefore be wary when predicting polymer data, since the intra-polymer (inter-sample) variation can be very large. When a polymer is only characterised by a small number of samples, we have no information on the range of intra-polymer variation present.

7.4 Modelling polymer properties

In order to use machine learning tools to build models to predict polymer properties the data must first be selected and converted into an appropriate format. The modelling component of PIKS used the data from the PoLyInfo database for this purpose. An overview of the complete process is described below followed by a detailed account.

7.4.1 Overview

The data was collected from the PoLyInfo database as XHTML via a web-spider (see Section 5.2.1). The XHTML was converted into XML using a custom parser (Section 5.2.2). From the XML the PoLyInfo formulae were extracted using XQuery (Section 5.5.1). The PoLyInfo formulae were converted into pentamers (See Section 7.4.3) represented in PML using the PoLyInfo reader (Section 5.2.4). The PML was then converted into CML molecules using JUMBO (Section 3.7) before being converted to SMILES[10] strings. SMILES was necessary as in interchange format to load the data into the Molecular Operating Environment (MOE) [61] which was used to calculate descriptors. The calculated descriptors were combined with the XML property data to produce a spreadsheet in CSV format of properties and descriptors using the XML2CSV tool (Section 5.3). This was then fed into a KNIME (Section 5.6) workflow for further processing followed by property prediction using the WEKA machine learning toolkit[33]. The full workflow is shown in (Figure 7.6)

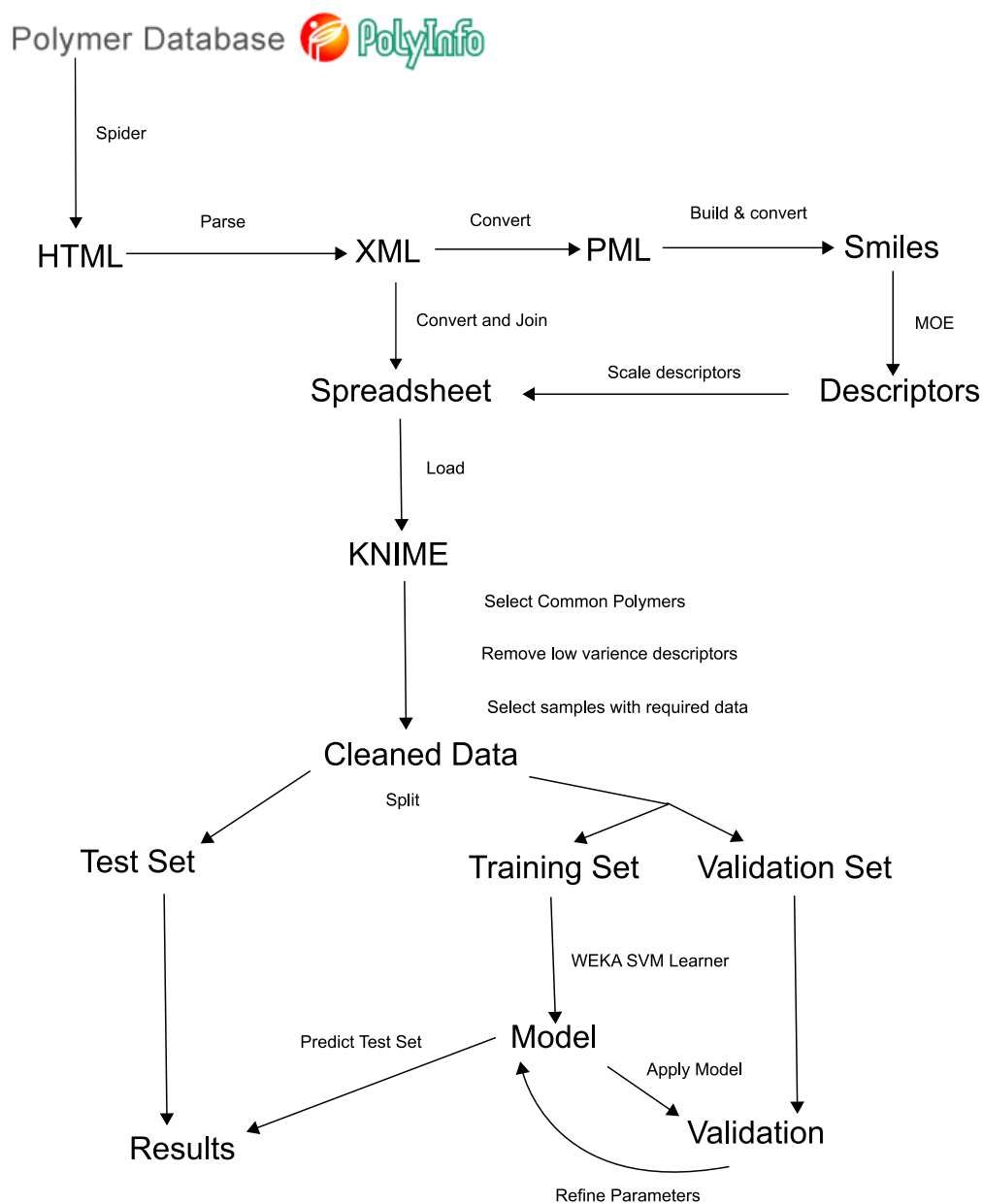


Figure 7.6: Workflow for the prediction

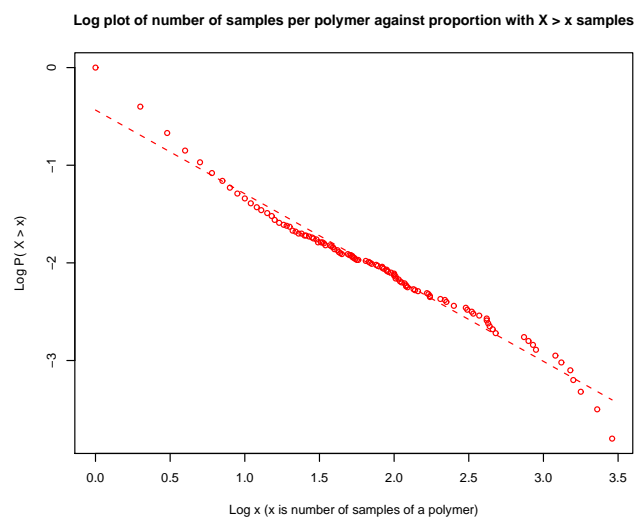


Figure 7.7: The powerlaw relationship between numbers of polymers with at least x samples, and the proportion of polymers with at least that many samples. There are many polymers with one sample and a few polymers with many samples.

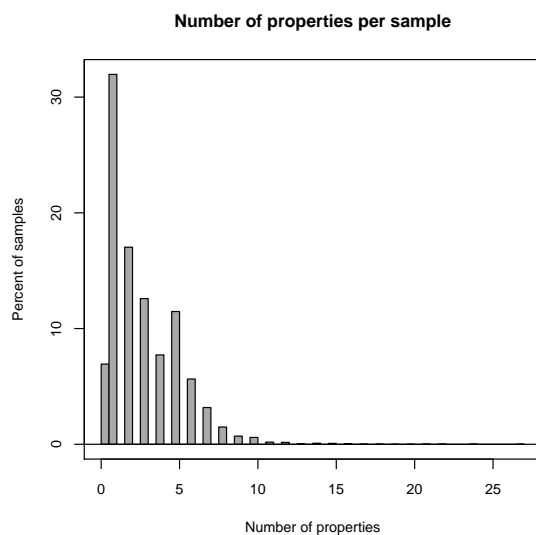


Figure 7.8: The distribution of number of properties per sample. The majority of the samples have either one or two properties

7.4.2 Data

Obtaining a reliable source of polymer data is of the utmost importance; without data to analyse there can be no analysis. For this purpose the PoLy-Info database (See Chapter 4) has been used. It consists of 13,402 polymers, with 262,396 property points taken from the literature. Each polymer has a specific number of samples in the database. The distribution of these samples approximately follows a Zipf's law[82] style power-law distribution[83] which can be seen in a cumulative distribution function plot (Figure 7.7). The extreme values represent the total number of polymers, and the number of samples of the most popular polymer. These two data points lie furthest from the straight-line. This is due to there being a very large number of polymers with only one sample, and the most popular polymer possessing a great number of samples. This shows that there are a small number of very common polymers and a large number of uncommon polymers, where how common a polymer is depends upon the number of times a sample of that polymer have appeared in a paper which has been added to the database.

Each sample of a polymer has between one and 27 properties. The distribution of properties per sample can be seen in Figure 7.8. This property distribution means that for any one sample we only have a limited number of properties. This creates difficulty in analysing the data as only limited information about any one sample is available. Obtaining information on the dataset (such as how many samples each polymer has) was done using XQuery. This method was chosen as, with the data held in an XML format, it was relatively fast to do so. For example, the XQuery required to extract the distribution of samples per polymer shown in Figure 7.8 was:

```
1 for $s in //Sample
2 return
3 <count>{count($s/Property)}</count>
```

The XQuery searches were performed using the eXist XML database software described in Chapter 5. Some difficulties were encountered with the size of the dataset since the default Java client is limited to only handle 10,000 nodes as the result of an XQuery. This necessitated loading the XQuery as a .xq file on the server and accessing the XML document which was produced.

Another workaround is to wrap the XQuery result in a container node which the Java client then regards as a result of only one node.

The data used was downloaded as HTML from the PoLyInfo database using the spider described in Section 5.2.1. This was then processed into XML (Section 5.2.2) before being converted into CSV format (Section 5.3). The CSV file was then loaded into a KNIME (Section 5.6) workflow which was used to filter; process; split into training, test and validation sets; and run the machine learning tools.

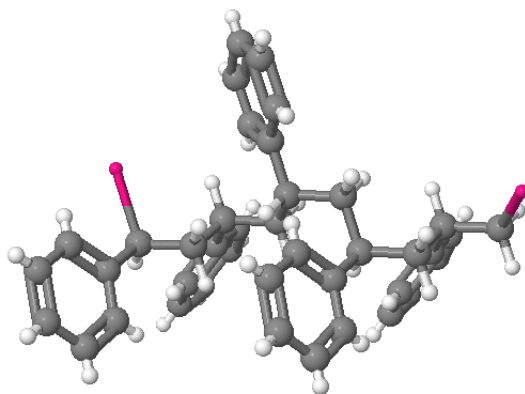
MOE[61] (Molecular Operating Environment) was used to calculate a range of 2D descriptors. These are described in Section 7.4.4.

7.4.3 Repeat units

The property data in the database is relatively straightforward to parse, but the PoLyInfo database uses a proprietary format for storing the connection table of the polymer (Section 4.6). This PoLyInfo formula was converted into PML using the program described in Section 5.2.4. The PML was then used to construct *pentamers* of the polymer. These pentamers consist of five repeat units joined together capped with hydrogen atoms.

The reason pentamers was chosen rather than a fully atomistic macro-molecule was due to size constraints on the molecules which could be processed by MOE. The MOE descriptor calculation software has hard limits on the number of atoms in a molecule for the calculation of descriptors. Some of the descriptors cannot be used on molecules with more than 250 atoms and MOE cannot represent a molecule larger than 1000. Using just a single repeat unit would give the capping end groups a large degree of influence on the molecule. Pentamers were chosen as the largest number of repeat units which would avoid hitting the atom limit for the larger repeat units. An example pentamer of poly(styrene) is shown in Figure 7.9. The PML template used to produce these pentamers is given below:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <fragment xmlns:p="http://www.xml-cml.org/mols/polyinfo"
3     xmlns="http://www.xml-cml.org/schema">
4     <fragment>
5         <molecule ref="p:fr"/>
```



Jmol

Figure 7.9: An example pentamer of poly(styrene) produced by the polymer builder before being loaded into MOE.

```

6      </fragment>
7      <join order="1" moleculeRefs2="PREVIOUS NEXT" atomRefs2="r1 r1">
8          <torsion>180</torsion>
9      </join>
10     <fragment countExpression="*(5)">
11         <join order="1" moleculeRefs2="PREVIOUS NEXT" atomRefs2="r2 r1">
12             <torsion>180</torsion>
13         </join>
14         <fragment id="placeholder">
15             </fragment>
16         </fragment>
17         <join order="1" moleculeRefs2="PREVIOUS NEXT" atomRefs2="r2 r1">
18             <torsion>180</torsion>
19         </join>
20         <fragment>
21             <molecule ref="p:fr"/>
22         </fragment>
23     </fragment>

```

To generate the pentamers, first the `<fragment>` element with `id="placeholder"` was replaced by the repeat unit of the polymer in question, and then the PML was expanded into CML using JUMBO. This process was repeated for each polymer in the PoLyInfo database for which fragments had been successfully transcribed.

7.4.4 Descriptors

The descriptors were calculated using MOE. The pentamers were loaded as a file of SMILES strings. These were then converted into 3D structures in MOE. There was a problem with the auto-generation of 3D structure by MOE. The initial layout of a handful of molecules which contained aromatic rings had a bond going through the centre of the ring. Using geometry optimisation would not free the bond from the local minimum. This necessitated adjusting torsion angles by hand. If this was not performed, then MOE would crash when calculating the descriptors, even when only calculating the 2D descriptors.

The descriptors were then calculated for each of the pentamers. The descriptors in MOE are divided into three classes:

- 2D** 2D descriptors only use the atoms and connection information of the molecule for the calculation. 3D coordinates and individual conformations are not considered
- i3D** Internal 3D descriptors use 3D coordinate information about each molecule; however, they are invariant to rotations and translations of the conformation
- x3D** External 3D descriptors also use 3D coordinate information but also require an absolute frame of reference (e.g. molecules docked into the same receptor)

The x3D descriptors are not appropriate, as they involve an absolute frame of reference which is not applicable to pentamers of different topologies and lengths. The i3D descriptors encode information in the internal three-dimensional structure of the pentamer. Since this is not representative of the conformation of a segment of a macromolecule in a polymer sample, these descriptors were not calculated either.

The 2D descriptors are generated from the connection table of the molecule. This information is independent of the conformation of the macromolecule. Ideally to represent an ensemble of macromolecules you could generate a set

of representative macromolecules from the ensemble and enumerate descriptors over this set. However the MOE package has a hard coded limit on the size of a molecule for descriptor calculation which is why pentamers were used instead. As well as these repeat unit based descriptors, the molecular weight of the sample and the polydispersity were included in the modelling process in order to represent sample variation. The MOE descriptors are described in detail in Appendix B.

7.4.5 Filtering the Data

The totality of data in the PoLyInfo database is not suitable for use directly for machine learning. As mentioned in Section 1.1.6 the presence of additives in a polymer sample will change the properties of that sample. For this reason any sample which was listed with an additive was removed from the dataset. Modelling the effect of additives on polymer properties is outside the scope of this work. In addition the data which was reported in Section 4.16 as problematic were also excluded.

After converting the data into a CSV file, the data were loaded into KNIME. Filtering of the polymers and descriptors was then carried out. As 181 descriptors were calculated it would be undesirable to use all of them for machine learning. The KNIME workflow normalised the descriptor values to be between -1 and +1. The descriptor `weinerPath` [26] had its log taken first, since it varied over such a large range of values. Part of the KNIME workflow removed all the descriptors which had a variance of less than 0.02. This resulted in 152 descriptors remaining. The polymers were filtered so that only polymers with at least 30 samples were used. This was to prevent the use of sparsely populated polymers with only a handful of datapoints. This reduced the number of polymers from 6171 with 30,291 total samples to 85 polymers with 17,395 total samples.

To build a model for a particular property, the data was again filtered to only contains samples with the property in question. As shown in Figure 4.10 most samples have only 1 or 2 properties. This means there is only a small subsection of samples which contain any two given properties. For example,

1,091 samples have both a glass transition temperature, a M_w and a M_w/M_n while 4,388 samples have a glass transition temperature. If a sample had two properties out of M_w , M_n and M_w/M_n the third was calculated.

The data was randomly split into a training, validation and test set. 60% of the data was selected to be the training set, 20% for the validation set and the remaining 20% for the test set. The reason for a different training and validation set is to allow parameter tweaking without risking information about the test set leaking into the model parameters [84].

7.4.6 Support Vector Machine Modelling of Properties

Machine learning models to predict polymer properties were built using the WEKA[33] SVM tool [85]. Support vector regression (SVR)[84] is an extension of Support Vector Machine (SVM)[86] classification.

A support vector machine is a method of machine learning which classifies data into two classes based on a separating hyperplane of maximal margin[87]. Regression is performed by constructing a hyperplane such that the distance of a point from the plane gives the predicted value. Support vector regression was conducted using the KNIME WEKA SVMreg node. The kernel used was RBFKernel. The regression optimiser used was RegSMOImproved [88] with default settings.

Glass Transition Temperature

Glass transition temperature (T_g) is an important polymer property as the physical properties of a polymer change if the temperature is above or below the T_g . The glass transition temperature is the most recorded property in the PoLyInfo database. The glass transition temperature gives us the greatest number of datapoints for model building, which is why this property was selected.

A range of models were produced varying the C and γ model parameters (described below) to obtain the best prediction of the validation set. Once this had been obtained, a model with these parameters was generated from the training and validation data and used to make predictions on the test

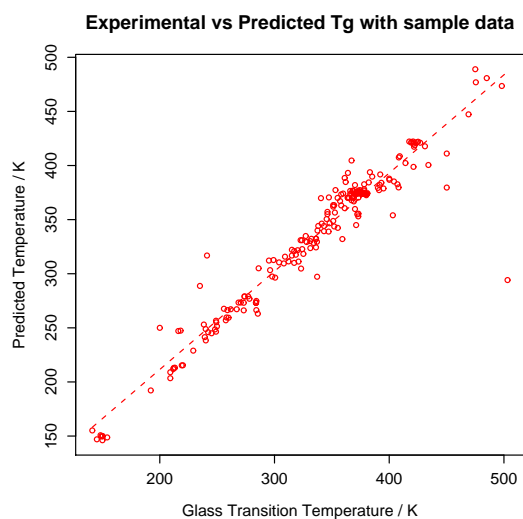
data. The parameters which maximised the correlation coefficient on the validation set were found to be $C = 1000$ and $\gamma = 5$. The C parameter controls the trade-off between closely fitting the training data, and the generality of the model. The γ parameter controls the shape of the radial basis function kernel.

The models were built both with the sample properties, M_w and M_w/M_n included in the descriptor set, and without. The model built without the sample properties can only predict one value per polymer, as it has no information about the individual samples. The performance of the models can be seen in Figures 7.10 and 7.11. From Figure 7.10(a) we can see that while a great number of samples are predicted to a high degree of accuracy (83% of samples predicted to within 5% error), there are some outliers with a large margin of error. The model with the sample properties had an $r^2 = 0.9593$ with a Root Mean Square (RMS) error of 20.1K. The model without the sample properties had an $r^2 = 0.9637$ with an RMS error of 19.1K.

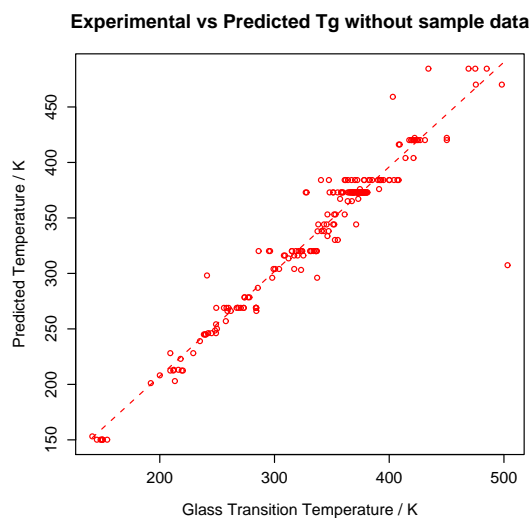
In Figure 7.10(b) we can see the predictions made by a model which has no information on the sample properties. In this case it can only predict one value for all samples of a particular polymer, which results in a spread of errors depending on the number of samples of that polymer, giving rise to the horizontal lines of data points in the figure. With this particular data set, the model without the sample data had a higher r^2 value.

Looking at the histograms in Figure 3.15 we can see that the model with the sample data has a greater number of predictions with less than 5% error, but also a greater number of predictions with greater than 20% error. The net effect is that it has a slightly worse r^2 correlation coefficient.

With a less noisy data set, the model with the sample characterisation data could achieve better results. The model without the sample characterisation data cannot improve by very much, since there will always be some error which depends on the intra-polymer variation which it is unable to account for. The proportion of samples which contain sample characterisation data is only 25% which limits the number of samples available for analysis.

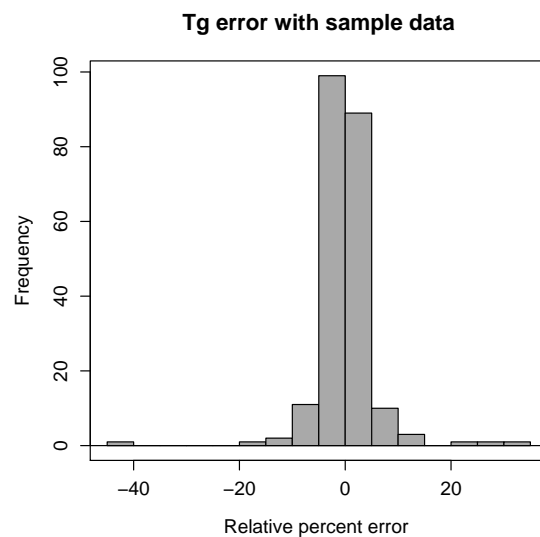


(a) Predicted vs experimental for T_g with sample data. Correlation coefficient $r^2 = 0.9593$, RMSE=20.1K

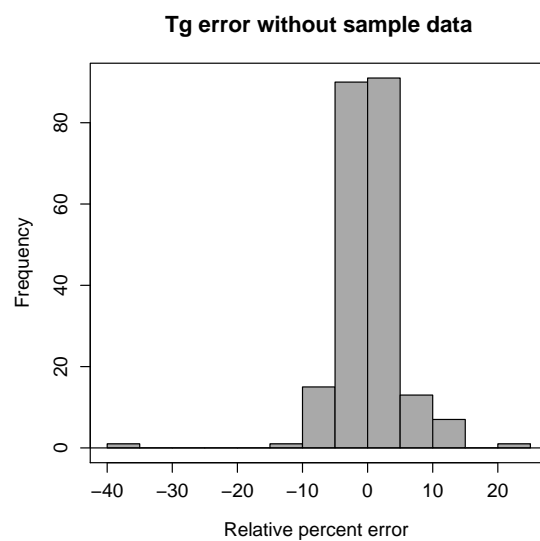


(b) Predicted vs experimental for T_g without sample data. Correlation coefficient $r^2 = 0.9637$, RMSE=19.1K

Figure 7.10: Plots of predicted and actual glass transition temperatures for SVR models with and without sample characterisation data of molecular weight and polydispersity. The model without sample data can only predict one value for each polymer, giving rise to the horizontal lines.



(a) Histogram of relative error with sample data



(b) Histogram of relative error without sample data

Figure 7.11: Histograms of the relative percentage error in the predictions of SVR models for glass transition temperature both with (7.11(a)) and without (7.11(b)) sample characterisation data of molecular weight and polydispersity.

7.5 Conclusions

The prediction of polymer properties on a per-sample basis is challenging. Existing work, [77] using only repeat units structure of polymers with no sample characterisation data, predicted glass transition temperature using computational neural networks with a prediction set RMS error of 21.9K ($r^2 = 0.96$). This is comparable to both the RMS error of 19.1K ($r^2 = 0.96$) obtained from the SVR model without sample properties, and the SVR model with sample properties of RMS error of 20.1K ($r^2 = 0.96$).

Another existing model is Radial Basis Function (RBF) neural networks with a $r^2=0.9269$ [89]. In this case only “high-molecular weight” samples were used, with the data taken from Cao ($r^2 = 0.9056$) [78] which in turn took it from Katritzky ($r^2 = 0.935$) [90] which cites Bicerano[91]. It is not clear which of Bicerano’s datasets has been used or what exactly a “high-molecular weight” sample consists of. The r^2 values of these approaches is slightly lower than that of the SVR model in this work.

If no sample characterisation data is used, it is clear that there will be a maximum performance of predictive models which depends on the intra-polymer variation of the samples in the test set. For predictive models of glass transition temperature to be able to cope with intra-polymer variation some sample characterisation data is needed. Unfortunately this characterisation data is hard to come by as the majority (75%) of data points do not have a M_w and M_w/M_n . Even fewer have a crystallinity (12%) while virtually none specify a degree of branching. The fraction of samples with a M_w , M_w/M_n and a degree of crystallinity is only 2.4%. This leaves a very small pool of data which is fully characterised.

Using an SVR to predict glass transition temperature of samples of polymers was moderately successful, but for the dataset used performed no better by the common metrics of r^2 and RMS error than the model without any sample data. If more sample characterisation data was available then the performance could be improved. Without using sample characterisation data, it will be impossible to improve predictions of polymer samples due to the inherent intra-polymer variation in sample properties.

Chapter 8

Conclusions

In this work the challenges of working with polymer data have been discussed. Unlike in traditional chemoinformatics which deals with small molecules, polymers consist of ensembles of macromolecules. This leads to difficulties in representation as well as uncertainty in the actual contents of a sample of polymer. The variability in structure and composition introduce large amounts of noise into aggregate polymer data.

In Chapter 2 informatics techniques normally used on small molecules were applied to atomistic macromolecules. This allowed descriptors to be calculated for macromolecules and used for machine learning to predict cleaning efficacy of polymers. The results for one of the three types of soiling were very good, with ROC AUC in excess of 0.9. The CART model used revealed an important structural motif which was present in those polymers with a specific monomer. The further use of this monomer in polymers for cleaning this soil type is therefore an area for future investigation.

In Chapter 3 the use of PML to represent polymers in a Markov-like process was discussed. An novel extension to PML which allows for non-Markov chains to be produced was introduced. This extension allows polymer samples, which have a distribution of repeat units that vary with molecular weight, to be described. The majority of polymer samples are not sufficiently characterised to make comparisons with.

In Suárez et al.[40] sufficiently well characterised samples were produced

to allow a comparison. Although the calculated distributions of ethylene % did not perfectly match the experimental data, the increasing trend with molecular weight was observed. The molecular weight distribution was similar to the experimental distribution. This new representational method allows for a better representation of a polymer sample, than merely a repeat unit with a M_w and M_w/M_n value.

In Chapter 4 the PoLyInfo database was investigated. While the documentation for the database indicates that there is an IUPAC structure-based name component to generate names; the existence of errors found in Chapter 6 between the structure and the name of a polymer shows that, either there are errors in the naming component, or it is not used in all occasions. The list of which units are allowed for which properties in the documentation was also found to be in error. In Appendix A.3 a listing of the units used for different properties is displayed. This does not always match the list of allowed units in the documentation for the database. It is therefore necessary to survey the actual data provided by sources rather than relying on documentation which may be out of date or incomplete.

The data in the PoLyInfo database is diverse and from many heterogeneous sources. The historical relationship found between T_m and T_g :

$$T_g \approx 0.7T_m$$

was found on small datasets, of up to 132 data points. The relationship has now been shown to still apply when calculated over the much larger 3,414 data points from the PoLyInfo dataset. This shows that the relationship holds for differing samples of polymers, as well as for different polymer types.

In Chapter 5 the components of the Polymer Informatics Knowledge System (PIKS) system were presented. This informatics toolkit has been developed for the performing informatics on polymer data. It was found to be capable of parsing polymer and monomer data from a variety of sources, storing the information as XML. This XML could be interrogated using XQuery to obtain the results of any query that might be required. A selection of broad XQueries providing different views on the data are shown in Appendix

A.

The reliability of data sources is of utmost importance. In Chapter 6 the level of errors in the monomer, polymer and reaction data in the PoLyInfo database was assessed. The monomers had a 5.8% error rate, the polymers a 7.3% error rate and the reactions a 2.9% error rate. This means that when accessing data from the database, there is an error in the chemical entity being looked at on top of experimental error in the physical data and uncertainties in the characterisation of polymer samples. The total uncertainty is therefore rather large, which poses problems for re-use of the data if it is not subjected to automatic validation.

Finally in Chapter 7 the PoLyInfo data was used to investigate if including sample characterisation data could improve predictions of glass transition temperature using Support Vector Regression. Although the SVR was comparable to previous methods of predicating polymer properties, achieving an $r^2 = 0.9593$ with a Root Mean Square (RMS) error of 20.1K with sample properties, and $r^2 = 0.9637$ with an RMS error of 19.1K without.

In this case it appears that adding the sample characterisation data did not help the prediction. Polymer sample variation depends on factors other than M_w and M_w/M_n , such as crystallinity and degree of branching. As fully characterised samples are not frequent enough to use for machine learning this presents a problem. If the accuracy of prediction is to improve, it is necessary to account for the variation within samples of the same polymer, but without adequate data it is impossible to make accurate predictions. Ideally descriptors representing the degree of branching of a polymer and the tacticity would be used to improve the prediction of sample variation. However the lack of full characterisation data for polymer samples makes this difficult. If the full GPC data for a polymer were available, descriptors based on the full molecular weight distribution could be used rather than the simple averages of M_w and M_w/M_n .

In this work a Polymer Informatics Knowledge System (PIKS) has been created which has facilitated the extraction, analysis, validation and prediction of polymer data. While this work has focused on the PoLyInfo database, with some data from JST, if a different spider component was written data

from any other data source could be included in addition, or instead. In addition a corpus of validated polymer reaction data in CML has been produced, which could be of use in further studies.

Appendix A

XQueries

A.1 Monomer ID listings by class

This XQuery returns a list sorted by monomer class of all the monomer names and PoLyInfo monomer IDs.

A.1.1 XQuery

```
1 declare namespace cml="http://www.xml-cml.org/schema";
2 <MonomerClasses>
3   {for $c in distinct-values(/cml:molecule/classMembership) return
4     <Class id="{ $c }">
5       { for $m in /cml:molecule[classMembership=$c] return
6         <Monomer
7           id="{string($m/cml:identifier[@convention="Polyinfo_Monomer_ID"]/@value)}">
8           {string($m/cml:name[not(@dictRef)])}</Monomer>
9         }
10      </Class>
11    }
12 </MonomerClasses>
```

A.1.2 Results

The complete results give 22,617 lines of XML. This is due to the fact that a lot of monomers are members of more than one class, so the resulting number of entries when sorted by class is larger than the number of monomers. Shown below is an abridged version as a demonstration, since the full listings go on for over 100 pages.

```
<MonomerClasses>
  <Class id="Dihalides">
```

```

<Monomer id="M3220864">1,4-dibromo-2,3,5,6-tetramethylbenzene</Monomer>
<Monomer id="M3220492">1,4-bis(dodecyloxy)-2,5-diiodobenzene</Monomer>
<Monomer id="M3231170">2-chloro-5-(4-fluorophenylsulfonyl)thiophene</Monomer>
<Monomer id="M3231002">4,4'-[(phenylimino)methylene]bischlorobenzene</Monomer>
<Monomer id="M3232216">naphthoquinone-1,4-bis(chlorimide)</Monomer>
<Monomer id="M2870233">dichloro(methyl)(pentafluorophenyl)silane</Monomer>
<Monomer id="M3230901">dichloro(butyl)(pentyl)silane</Monomer>
<Monomer id="M3232113">2,6-bis[(4-fluorophenyl)sulfonyl]naphthalene</Monomer>
<Monomer id="M2850552">1,4-bis(bromomethyl)benzene</Monomer>
<Monomer id="M2820602">1,4-dibromo-2-fluorobenzene</Monomer>
<Monomer id="M2870466">1,4-bis(chlorodimethylsilyl)benzene</Monomer>
<Monomer id="M2860371">2,5-dichloropiperazine</Monomer>
<Monomer id="M3232233">butyl(dichloro)(phenethyl)silane</Monomer>
<Monomer id="M2570231">dichlorodisulfane</Monomer>
<Monomer id="M2871400">1,7-dichloroheptane</Monomer>
<Monomer id="M3220876">1,4-bis(chlorobutyl)benzene</Monomer>
<Monomer id="M2132849">manganese(II) bis(2,4,6-trichlorophenolate)</Monomer>
<Monomer id="M0632754">1-[4-(2,2-dibromovinyl)phenyl]-2-phenylacetylene</Monomer>
<Monomer id="M2850564">bis(3-chlorophenyl) ether</Monomer>
<Monomer id="M2850600">dichlorodifluoromethane</Monomer>
.....
</Class>
</MonomerClasses>

```

A.2 Properties and Experimental Methods

This XQuery returns a list of Physical Properties recorded in the Database with as a list of properties with a sublist as the different experimental methods used to record them.

A.2.1 XQuery

```

1 <Properties>
2   {for $p in distinct-values(//Property[Data/@id="Method"]/@id) return
3     <Property id="{ $p }">
4     { for $m in distinct-values(//Data[@id="Method" and parent::*/@id=$p]) return
5       <Method>{ $m }</Method>
6     }
7   </Property>
8 }
9 </Properties>

```

A.2.2 Results

This query produces several hundred pages of output, since there are a great deal of different ways each method type have been reported where they vary

slightly in the way they are written. An abridged output is shown below as an example.

```

<Properties>
  <Property id="Thermal decomposition temp.">
    <Method>TG</Method>
    <Method>DTA</Method>
    <Method>Dilatometry</Method>
    <Method>DSC</Method>
    <Method>Wt. loss</Method>
    <Method>TG-DTA</Method>
    <Method>TG , DSC</Method>
    <Method>Simultaneous thermal anal. (STA)</Method>
    <Method>Isothermogravimetric analysys</Method>
  .....
    <Method>TG-MS</Method>
  </Property>
  <Property id="Glass transition temp.">
    <Method>TMA</Method>
    <Method>Dilatometry</Method>
    <Method>DSC</Method>
    <Method>DMA</Method>
    <Method>Ultrasonic Velocity Measurement</Method>
    <Method>Modulated temp. DSC (MTDSC)</Method>
    <Method>DTA</Method>
    <Method>Gas chromatgraphy method</Method>
    <Method>[NR]</Method>
    <Method>Dielectric relaxation</Method>
    <Method>Adiabatic vacuum calorimeter</Method>
    <Method>Fluorescence spectrophotometer</Method>
    <Method>PVT</Method>
    <Method>Cp measurement</Method>
  .....
    <Method>Photoelastic coeff.</Method>
  </Property>
  <Property id="Melting temp.">
    <Method>Microscope</Method>
    <Method>DSC</Method>
    <Method>Hot stage polarizing microscope</Method>
    <Method>Capillary</Method>
    <Method>DTA</Method>
    <Method>[NR]</Method>
    <Method>Dilatometry</Method>
    <Method>TG</Method>
    <Method>DMA</Method>
  ....
    <Method>TMA</Method>
  </Property>
  ....
</Properties>

```


A.3 Units

The PoLyInfo database has some documentation which claims to list all the units used for each property in the database, along with the preferred unit. However upon inspection of the data it became apparent that there are many more units in the actual data than listed in the documentation. The following XQuery will retrieve the units used for each property.

A.3.1 XQuery

```
1 <Properties>
2   {for $p in distinct-values(//Property[Data/@id="Method"]/@id) return
3     <Property id="{ $p }">
4     { for $m in distinct-values(//Property/@unit[parent::*/@id=$p]) return
5       <Unit>{ $m }</Unit>
6     }
7   </Property>
8 }
9 </Properties>
```

A.3.2 Results

This produces a manageable length of output, so the results listing is in full. Where a `<Unit/>` is shown, a sample existed with that property where the unit could not be parsed for some reason, resulting in an empty unit. The listing of `<Unit>[]</Unit>` represents the case where the unit is of zero dimensionality (more specifically where a datapoint is recorded with a parsable value but reported with no units.)

```
<Properties>
  <Property id="Thermal decomposition temp.">
    <Unit/>
    <Unit>[C]</Unit>
    <Unit>[F]</Unit>
    <Unit>[K]</Unit>
  </Property>
  <Property id="Tensile properties">
    <Unit/>
  </Property>
  <Property id="Glass transition temp.">
    <Unit>[C]</Unit>
    <Unit/>
    <Unit>[K]</Unit>
    <Unit>[]</Unit>
  </Property>
  <Property id="Melting temp.">
    <Unit>[C]</Unit>
```

```

        <Unit/>
        <Unit>[K]</Unit>
        <Unit>[F]</Unit>
        <Unit>[]</Unit>
    </Property>
    <Property id="Gas permeability and diffusion">
        <Unit/>
    </Property>
    <Property id="Surface tension">
        <Unit>[mN/m]</Unit>
        <Unit>[dyn/cm]</Unit>
        <Unit/>
        <Unit>[]</Unit>
        <Unit>[mJ/m2]</Unit>
        <Unit>[erg/cm2]</Unit>
    </Property>
    <Property id="Density">
        <Unit>[g/cm3]</Unit>
        <Unit/>
    </Property>
    <Property id="Intrinsic viscosity [eta]">
        <Unit>[dl/g]</Unit>
        <Unit/>
        <Unit>[]</Unit>
        <Unit>[l/cc]</Unit>
    </Property>
    <Property id="Electric resistivity">
        <Unit>[1/(ohm*cm)]</Unit>
        <Unit/>
        <Unit>[1/(ohm*cm)1/(ohm*cm)]</Unit>
    </Property>
    <Property id="Thermal expansion">
        <Unit/>
    </Property>
    <Property id="Interfacial tension">
        <Unit>[dyn/cm]</Unit>
        <Unit/>
        <Unit>[mN/m]</Unit>
    </Property>
    <Property id="Heat of fusion">
        <Unit>[J/g]</Unit>
        <Unit>[cal/g]</Unit>
        <Unit/>
        <Unit>[kJ/kg]</Unit>
        <Unit>[]</Unit>
        <Unit>[cal/cc]</Unit>
        <Unit>[cal/cm3]</Unit>
        <Unit>[J/mol]</Unit>
        <Unit>[J/cc]</Unit>
        <Unit>[erg/cm3]</Unit>

```

```

        <Unit>[J/cm3]</Unit>
        <Unit>[kcal/g]</Unit>
    </Property>
    <Property id="Crystallization">
        <Unit/>
    </Property>
    <Property id="Sedimentation coefficient">
        <Unit>[svedberg(1E(-13)second)]</Unit>
        <Unit>[second]</Unit>
        <Unit/>
    </Property>
    <Property id="Diffusion coefficient">
        <Unit>[cm2/s]</Unit>
        <Unit/>
    </Property>
    <Property id="Solubility parameter">
        <Unit>[(J/cm3)1/2]</Unit>
        <Unit>[(cal/cm3)1/2]</Unit>
        <Unit/>
        <Unit>[(MPa)1/2]</Unit>
    </Property>
    <Property id="Theta-solvent/Theta-temp.">
        <Unit/>
    </Property>
    <Property id="Radius of gyration">
        <Unit/>
        <Unit>[nm]</Unit>
        <Unit>[]</Unit>
    </Property>
    <Property id="Second virial coefficient">
        <Unit/>
        <Unit>[cm3*mol/g2]</Unit>
        <Unit>[bar*cm6/g2]</Unit>
        <Unit>[cm3/g2]</Unit>
        <Unit>[atm*cm6/g2]</Unit>
        <Unit>[(ml/g)**2]</Unit>
        <Unit>[mol/cm3]</Unit>
        <Unit>[atm*ml**2/g**2]</Unit>
        <Unit>[ml/g2]</Unit>
    </Property>
    <Property id="Softening points">
        <Unit/>
    </Property>
    <Property id="Dielectric constant (AC)">
        <Unit>[]</Unit>
    </Property>
    <Property id="Melt viscosity">
        <Unit>[N*s/m2(=Pa*s)]</Unit>
        <Unit/>
        <Unit>[dyn*s/cm2(=poise)]</Unit>

```

```

        <Unit>[]</Unit>
        <Unit>[cSt]</Unit>
        <Unit>[P]</Unit>
    </Property>
    <Property id="Heat conduction (Heat transfer)">
        <Unit/>
    </Property>
    <Property id="Specific heat capacity">
        <Unit/>
    </Property>
    <Property id="Dynamic flexural modulus">
        <Unit/>
    </Property>
    <Property id="Dielectric dispersion">
        <Unit/>
    </Property>
    <Property id="Radiation resistance">
        <Unit/>
    </Property>
    <Property id="Refractive index">
        <Unit>[]</Unit>
    </Property>
    <Property id="Dynamic tensile properties">
        <Unit/>
    </Property>
    <Property id="G value">
        <Unit>[events/100ev]</Unit>
        <Unit/>
        <Unit>[]</Unit>
    </Property>
    <Property id="PVT relation">
        <Unit/>
    </Property>
    <Property id="Dielectric properties">
        <Unit/>
    </Property>
    <Property id="Crystallization kinetics">
        <Unit/>
    </Property>
    <Property id="Dynamic shear modulus">
        <Unit/>
    </Property>
    <Property id="Dynamic compressive properties">
        <Unit/>
    </Property>
    <Property id="Izod impact">
        <Unit>[ft*lb/in]</Unit>
        <Unit>[kg*cm/cm]</Unit>
        <Unit/>
        <Unit>[kJ/m]</Unit>
    </Property>

```

```

        <Unit>[kg/cm2]</Unit>
        <Unit>[J/m]</Unit>
        <Unit>[kJ/m2]</Unit>
        <Unit>[Pa]</Unit>
        <Unit>[kgcm/cm2]</Unit>
        <Unit>[mJ/mm2]</Unit>
    </Property>
    <Property id="Compressive properties">
        <Unit/>
    </Property>
    <Property id="LC phase transition temp.">
        <Unit>[C]</Unit>
        <Unit/>
        <Unit>[K]</Unit>
        <Unit>[]</Unit>
    </Property>
    <Property id="Water absorption">
        <Unit/>
        <Unit>[wt%]</Unit>
        <Unit>[mg/cm2]</Unit>
        <Unit>[other]</Unit>
        <Unit>[]</Unit>
        <Unit>[mg/g]</Unit>
        <Unit>[H2O/SO3H(mol)]</Unit>
    </Property>
    <Property id="Stress-optical coefficient">
        <Unit>[Brewster]</Unit>
        <Unit>[cm2/dyn]</Unit>
        <Unit>[1/(GPa)]</Unit>
        <Unit>[m2/N]</Unit>
        <Unit>[psi/(fringe*in.)]</Unit>
        <Unit/>
        <Unit>[cm2/kg]</Unit>
        <Unit>[]</Unit>
    </Property>
    <Property id="Shear properties">
        <Unit/>
    </Property>
    <Property id="Flexural properties">
        <Unit/>
    </Property>
    <Property id="Charpy impact">
        <Unit>[kJ/m2]</Unit>
        <Unit/>
        <Unit>[kg*cm/cm2]</Unit>
        <Unit>[J/m]</Unit>
    </Property>
    <Property id="Dynamic viscosity">
        <Unit/>
    </Property>

```

```

<Property id="Bulk modulus">
  <Unit/>
  <Unit>[GPa]</Unit>
  <Unit>[MPa]</Unit>
</Property>
<Property id="Brittleness temp.">
  <Unit>[F]</Unit>
  <Unit>[C]</Unit>
  <Unit/>
</Property>
<Property id="Shore hardness">
  <Unit>[]</Unit>
</Property>
<Property id="Vicat softening point">
  <Unit/>
</Property>
<Property id="UL Flammability code rating">
  <Unit>[]</Unit>
  <Unit/>
</Property>
<Property id="Oxygen Index">
  <Unit>[%]</Unit>
  <Unit/>
</Property>
<Property id="Deflection temperature under load (HDT)">
  <Unit>[F]</Unit>
  <Unit>[C]</Unit>
  <Unit>[K]</Unit>
  <Unit/>
</Property>
<Property id="Tensile creep">
  <Unit/>
</Property>
<Property id="Water vapor transmission">
  <Unit>[g/(m2*d*bar)]</Unit>
  <Unit>[g*mil/(cm2*24h)]</Unit>
  <Unit>[g/(m2*h)]</Unit>
  <Unit>[g/(cm2*24h)]</Unit>
  <Unit/>
  <Unit>[g*mil/(100in.**2*24h*44mmHg)]</Unit>
  <Unit>[g*mm/(m2*24h*atm)]</Unit>
</Property>
<Property id="Flexural creep">
  <Unit/>
</Property>
<Property id="Rockwell hardness">
  <Unit>[L-]</Unit>
  <Unit>[R-]</Unit>
  <Unit>[M-]</Unit>
  <Unit>[J-]</Unit>

```

```
        <Unit/>
    </Property>
    <Property id="UL Temp. index">
        <Unit/>
    </Property>
</Properties>
```

Appendix B

Descriptors

The following descriptors were used by the MOE package in Chapter 7. These descriptions are taken from the MOE manual.

B.1 Physical

The following descriptors are described as *physical*:

apol Sum of the atomic polarizabilities (including implicit hydrogens) with polarizabilities taken from [92].

bpol Sum of the absolute value of the difference between atomic polarizabilities of all bonded atoms in the molecule (including implicit hydrogens) with polarizabilities taken from [92].

density Molecular mass density: Weight divided by vdw_vol (amu/Å³).

FCharge Total charge of the molecule (sum of formal charges).

mr Molecular refractivity (including implicit hydrogens). This property is calculated from an 11 descriptor linear model [93] with $r^2 = 0.997$, RMSE = 0.168 on 1,947 small molecules.

SMR Molecular refractivity (including implicit hydrogens). This property is an atomic contribution model [94] that assumes the correct protonation state (washed structures). The model was trained on 7000 structures and results may vary from the mr descriptor.

Weight Molecular weight (including implicit hydrogens) in atomic mass units with atomic weights taken from [92].

logP(o/w) Log of the octanol/water partition coefficient (including implicit hydrogens). This property is calculated from a linear atom type model [95] with $r^2 = 0.931$, RMSE=0.393 on 1,827 molecules.

logS Log of the aqueous solubility (mol/L). This property is calculated from an atom contribution linear atom type model [96] with $r^2 = 0.90$, 1,200 molecules.

reactive Indicator of the presence of reactive groups. A non-zero value indicates that the molecule contains a reactive group. The table of reactive groups is based on the Oprea set [97] and includes metals, phospho-, N/O/S-N/O/S single bonds, thiols, acyl halides, Michael Acceptors, azides, esters, etc.

rsynth A value in [0,1] indicating the synthetic reasonableness, or feasibility, of the chemical structure. A value of 0 means it is unlikely that the molecule can be synthesized while a value of 1 means that it is likely that the molecule can be synthesized. The value reflects the fraction of heavy atoms in the molecule that can be traced back to starting materials fragments resulting from retrosynthetic disconnection rules.

SlogP Log of the octanol/water partition coefficient (including implicit hydrogens). This property is an atomic contribution model [94] that calculates logP from the given structure; i.e. the correct protonation state (washed structures). Results may vary from the logP(o/w) descriptor. The training set for SlogP was 7000 structures.

TPSA Polar surface area (\AA^2) calculated using group contributions to approximate the polar surface area from connection table information only. The parameterization is that of Ertl et al. [98].

vdw_vol van der Waals volume (\AA^3) calculated using a connection table approximation.

vdw_area Area of van der Waals surface (\AA^2) calculated using a connection table approximation.

B.2 Subdivided Surface Areas

The next set of descriptors are *subdivided surface areas*. For these a sum is taken over the set of atoms v_i which meet a condition. The property L_i is the contribution to logP(o/w) for the atom i calculated by the SlogP descriptor [94]. The property R_i is the contribution to molar refractivity for atom i calculated by the SMR descriptor [94]. The intervals are defined with

a ‘(’ when the lower limit is excluded from the interval and a ‘[’ when the lower limit is included in the interval.

SlogP_VSA0 Sum of v_i such that $L_i \leq -0.4$.

SlogP_VSA1 Sum of v_i such that L_i is in $(-0.4, -0.2]$.

SlogP_VSA2 Sum of v_i such that L_i is in $(-0.2, 0]$.

SlogP_VSA3 Sum of v_i such that L_i is in $(0, 0.1]$.

SlogP_VSA4 Sum of v_i such that L_i is in $(0.1, 0.15]$.

SlogP_VSA5 Sum of v_i such that L_i is in $(0.15, 0.20]$.

SlogP_VSA6 Sum of v_i such that L_i is in $(0.20, 0.25]$.

SlogP_VSA7 Sum of v_i such that L_i is in $(0.25, 0.30]$.

SlogP_VSA8 Sum of v_i such that L_i is in $(0.30, 0.40]$.

SlogP_VSA9 Sum of v_i such that $L_i > 0.40$.

SMR_VSA0 Sum of v_i such that R_i is in $[0, 0.11]$.

SMR_VSA1 Sum of v_i such that R_i is in $(0.11, 0.26]$.

SMR_VSA2 Sum of v_i such that R_i is in $(0.26, 0.35]$.

SMR_VSA3 Sum of v_i such that R_i is in $(0.35, 0.39]$.

SMR_VSA4 Sum of v_i such that R_i is in $(0.39, 0.44]$.

SMR_VSA5 Sum of v_i such that R_i is in $(0.44, 0.485]$.

SMR_VSA6 Sum of v_i such that R_i is in $(0.485, 0.56]$.

SMR_VSA7 Sum of v_i such that $R_i > 0.56$.

B.3 Atom Counts and Bond Counts

These descriptors are based on atom and bond counts. Here *heavy atoms* refer to any atom other than hydrogen, *trivial atoms* refer to a hydrogen bonded to only one other atom, h is the number of attached hydrogens to the atom if all remaining valences were bonded to hydrogen, d is the heavy degree which is the number of heavy atoms bonded to the atom in question and Z_i is the atomic number of atom i .

a_aro Number of aromatic atoms.

a_count Number of atoms (including implicit hydrogens). This is calculated as the sum of $(1 + h_i)$ over all non-trivial atoms i .

a_heavy Number of heavy atoms $\#\{Z_i | Z_i > 1\}$.

a_ICM Atom information content (mean). This is the entropy of the element distribution in the molecule (including implicit hydrogens but not lone pair pseudo-atoms). Let n_i be the number of occurrences of atomic number i in the molecule. Let $p_i = n_i/n$ where n is the sum of the n_i . The value of **a_ICM** is the negative of the sum over all i of $p_i \log p_i$.

a_IC Atom information content (total). This is calculated to be **a_ICM** times n .

a_nH Number of hydrogen atoms (including implicit hydrogens). This is calculated as the sum of h_i over all non-trivial atoms i plus the number of non-trivial hydrogen atoms.

a_nB Number of boron atoms: $\#\{Z_i | Z_i = 5\}$.

a_nC Number of carbon atoms: $\#\{Z_i | Z_i = 6\}$.

a_nN Number of nitrogen atoms: $\#\{Z_i | Z_i = 7\}$.

a_nO Number of oxygen atoms: $\#\{Z_i | Z_i = 8\}$.

a_nF Number of fluorine atoms: $\#\{Z_i | Z_i = 9\}$.

a_nP Number of phosphorus atoms: $\#\{Z_i | Z_i = 15\}$.

a_nS Number of sulfur atoms: $\#\{Z_i | Z_i = 16\}$.

a_nCl Number of chlorine atoms: $\#\{Z_i | Z_i = 17\}$.

a_nBr Number of bromine atoms: $\#\{Z_i | Z_i = 35\}$.

a_nI Number of iodine atoms: $\#\{Z_i | Z_i = 53\}$.

b_1rotN Number of rotatable single bonds. Conjugated single bonds are not included (e.g. ester and peptide bonds).

b_1rotR Fraction of rotatable single bonds: **b_1rotN** divided by **b_heavy**.

b_ar Number of aromatic bonds.

b_count Number of bonds (including implicit hydrogens). This is calculated as the sum of $(d_i/2 + h_i)$ over all non-trivial atoms i .

b_double Number of double bonds. Aromatic bonds are not considered to be double bonds.

b_heavy Number of bonds between heavy atoms.

b_rotN Number of rotatable bonds. A bond is rotatable if it has order 1, is not in a ring, and has at least two heavy neighbors.

b_rotR Fraction of rotatable bonds: **b_rotN** divided by **b_heavy**.

b_single Number of single bonds (including implicit hydrogens). Aromatic bonds are not considered to be single bonds.

b_triple Number of triple bonds. Aromatic bonds are not considered to be triple bonds.

chiral The number of chiral centers.

chiral_u The number of unconstrained chiral centers.

lip_acc The number of O and N atoms.

lip_don The number of OH and NH atoms.

lip_druglike One if and only if **lip_violation** < 2 otherwise zero.

lip_violation The number of violations of Lipinski's Rule of Five [99].

nmol The number of molecules (connected components).

opr_brigid The number of rigid bonds from [97].

opr_leadlike One if and only if **opr_violation** < 2 otherwise zero.

opr_nring The number of ring bonds from [97].

opr_nrot The number of rotatable bonds from [97].

opr_violation The number of violations of Oprea's lead-like test [97].

rings The number of rings.

VAdjMa Vertex adjacency information (magnitude): $1 + \log_2 m$ where m is the number of heavy-heavy bonds. If m is zero, then zero is returned.

VAdjEq Vertex adjacency information (equality): $-(1 - f) \log_2(1 - f) - f \log_2 f$ where $f = (n^2 - m)/n^2$, n is the number of heavy atoms and m is the number of heavy-heavy bonds. If f is not in the open interval (0,1), then 0 is returned.

B.4 Kier and Hall Connectivity and Kappa Shape Indices

These descriptors are based on the connectivity of the connection table. For each *heavy atom* i , $v_i = (p_i - h_i)/Z_i - p_i - 1$ where p_i is the number of s and p valence electrons. The definitions of h_i and d_i are as before, n is the number of non-hydrogen atoms in the molecule, m is the number of bonds between non-hydrogen atoms and a is the sum over all atoms of $(r_i/r_c - 1)$ where r_i is the covalent radius of atom i and r_c is the covalent radius of a carbon atom. Additional parameters are 2P and 3P for the number of paths of length 2 and 3 respectively.

chi0 Atomic connectivity index (order 0) from [31] and [30]. This is calculated as the sum of $1/\sqrt{d_i}$ over all heavy atoms i with $d_i > 0$.

chi0_C Carbon connectivity index (order 0). This is calculated as the sum of $1/\sqrt{d_i}$ over all carbon atoms i with $d_i > 0$.

chi1 Atomic connectivity index (order 1) from [31] and [30]. This is calculated as the sum of $1/\sqrt{d_i d_j}$ over all bonds between heavy atoms i and j where $i < j$.

chi1_C Carbon connectivity index (order 1). This is calculated as the sum of $1/\sqrt{d_i d_j}$ over all bonds between carbon atoms i and j where $i < j$.

chi0v Atomic valence connectivity index (order 0) from [31] and [30]. This is calculated as the sum of $1/\sqrt{v_i}$ over all heavy atoms i with $v_i > 0$.

chi0v_C Carbon valence connectivity index (order 0). This is calculated as the sum of $1/\sqrt{v_i}$ over all carbon atoms i with $v_i > 0$.

chi1v Atomic valence connectivity index (order 1) from [31] and [30]. This is calculated as the sum of $1/\sqrt{v_i v_j}$ over all bonds between heavy atoms i and j where $i < j$.

chi1v_C Carbon valence connectivity index (order 1). This is calculated as the sum of $1/\sqrt{v_i v_j}$ over all bonds between carbon atoms i and j where $i < j$.

Kier1 First kappa shape index: $n(n-1)^2/m^2$ [31].

Kier2 Second kappa shape index: $(n-1)(n-2)^2/({}^2P)^2$ [31].

Kier3 Third kappa shape index: $(n-1)(n-3)^2/({}^3P)^2$ for odd n , and $(n-3)(n-2)^2/({}^3P)^2$ for even n [31].

KierA1 First alpha modified shape index: $s(s-1)^2/m^2$ where $s = n + a$ [31].

KierA2 Second alpha modified shape index: $(s-1)(s-2)^2/(^2P)^2$ where $s = n + a$ [31].

KierA3 Third alpha modified shape index: $(s-1)(s-3)^2/(^3P)^2$ for odd n , and $(s-3)(s-2)^2/(^3P)^2$ for even n where $s = n + a$ [31].

KierFlex Kier molecular flexibility index: (KierA1) (KierA2) / n [31].

zagreb Zagreb index: the sum of d_i^2 over all heavy atoms i .

B.5 Adjacency and Distance Matrix Descriptors

This set of descriptors are calculated from the an adjacency and distance matrix. The matrix adjacency matrix M_{ij} has value 1 if atoms i and j are bonded, and 0 otherwise. The distance matrix D_{ij} has values equal to the shortest bond distance between atoms i and j .

balabanJ Balaban’s connectivity topological index [100].

BCUT_PEOE The BCUT descriptors [101] are calculated from the eigenvalues of a modified adjacency matrix. Each ij entry of the adjacency matrix takes the value $1/\sqrt{b_{ij}}$ where b_{ij} is the formal bond order between bonded atoms i and j . The diagonal takes the value of the PEOE partial charges. The resulting eigenvalues are sorted and the smallest, 1/3-ile, 2/3-ile and largest eigenvalues are reported.

BCUT_SLOGP The BCUT descriptors using atomic contribution to logP (using the Wildman and Crippen SlogP method) instead of partial charge.

BCUT_SMR The BCUT descriptors using atomic contribution to molar refractivity (using the Wildman and Crippen SMR method) instead of partial charge.

diameter Largest value in the distance matrix [102].

petitjean Value of (diameter - radius) / diameter.

GCUT_PEOE The GCUT descriptors are calculated from the eigenvalues of a modified graph distance adjacency matrix. Each ij entry of the adjacency matrix takes the value $1/\sqrt{d_{ij}}$ where d_{ij} is the (modified) graph distance between atoms i and j . The diagonal takes the value of the PEOE partial charges. The resulting eigenvalues are sorted and the smallest, 1/3-ile, 2/3-ile and largest eigenvalues are reported.

GCUT_SLOGP The GCUT descriptors using atomic contribution to logP (using the Wildman and Crippen SlogP method) instead of partial charge.

GCUT_SMR The GCUT descriptors using atomic contribution to molar refractivity (using the Wildman and Crippen SMR method) instead of partial charge.

petitjeanSC Petitjean graph Shape Coefficient as defined in [102]: (diameter - radius) / radius.

radius If r_i is the largest matrix entry in row i of the distance matrix D , then the radius is defined as the smallest of the r_i [102].

VDistEq If m is the sum of the distance matrix entries then VdistEq is defined to be the sum of $\log_2 m - p_i \log_2 p_i / m$ where p_i is the number of distance matrix entries equal to i .

VDistMa If m is the sum of the distance matrix entries then VDistMa is defined to be the sum of $\log_2 m - D_{ij} \log_2 D_{ij} / m$ over all i and j .

wienerPath Wiener path number: half the sum of all the distance matrix entries as defined in [29] and [26].

wienerPol Wiener polarity number: half the sum of all the distance matrix entries with a value of 3 as defined in [29].

Bibliography

- [1] B. Claus and D. Underwood. Discovery informatics: its evolving role in drug discovery. *Drug discovery today*, 7(18):957–966, 2002. ISSN 1359-6446.
- [2] C. Manly, S. Louise-May, and J. Hammer. The impact of informatics and computational chemistry on synthesis and screening. *Drug discovery today*, 6(21):1101–1110, 2001. ISSN 1359-6446.
- [3] M. Frenklach. Transforming data into knowledge—Process Informatics for combustion chemistry. *Proceedings of the Combustion Institute*, 31(1):125–140, 2007. ISSN 1540-7489.
- [4] N. Adams and P. Murray-Rust. Engineering Polymer Informatics: Towards the Computer-Aided Design of Polymers. *Macromolecular Rapid Communications*, 29(8):615–632, 2008. ISSN 1521-3927.
- [5] IUPAC. *Compendium of Chemical Terminology, 2nd ed. (the "Gold Book")*. Blackwell Scientific Publications, Oxford, 1997.
- [6] P. Flory. *Principles of Polymer Chemistry*. Cornell University Press, 1953.
- [7] A. D. Jenkins, P. Kratochvíl, R. F. T. Stepto, and U. W. Suter. Glossary of basic terms in polymer science (IUPAC recommendations 1996). *Pure and Applied Chemistry*, 68(12):2287–2311, 1996. ISSN 0033-4545.
- [8] M. P. Stevens. *Polymer Chemistry An Introduction*. Oxford University press, 1999.
- [9] A. Dalby, J. G. Nourse, W. D. Hounshell, A. K. I. Gushurst, D. L. Grier, B. A. Leland, and J. Laufer. Description of several chemical structure file formats used by computer programs developed at molecular design limited. *Journal of Chemical Information and Computer Sciences*, 32(3):244–255, May 1992.

- [10] D. Weininger. SMILES, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of Chemical Information and Computer Sciences*, 28(1):31–36, February 1988.
- [11] W. M. Brown, S. Martin, M. D. Rintoul, and J. Faulon. Designing novel polymers with targeted properties using the signature molecular descriptor. *Journal of Chemical Information and Modeling*, 46(2):826–835, March 2006.
- [12] I. Stott. Extending smiles to represent polymer repeat units. Unilever.
- [13] extensible mark up language. URL <http://www.w3.org/TR/2008/REC-xml-20081126/>.
- [14] S. Boag, D. Chamberlin, M. F. Fernández, D. Florescu, J. Robie, and J. Siméon. Xquery 1.0: An xml query language. *W3C Working Draft*, 7, 2001. URL <http://www.w3.org/TR/2007/REC-xquery-20070123/>.
- [15] P. Murray-Rust and H. Rzepa. Chemical markup, XML, and the world-wide web. 1. basic principles. *J. Chem. Inf. Comput. Sci*, 39(6):928–942, 1999.
- [16] P. Murray-Rust and H. Rzepa. Chemical markup, XML and the world-wide web. 2. information objects and the CMLDOM. *J. Chem. Inf. Comput. Sci*, 41(5):1113–1123, 2001.
- [17] P. Murray-Rust and H. S. Rzepa. Chemical markup, XML and the worldwide web part 4: CML schema. *Journal of Chemical Informatics and Computer Science*, 43(4):757–772, 2003.
- [18] G. Holliday, P. Murray-Rust, and H. Rzepa. Chemical markup, XML, and the world wide web. 6. CMLReact, an XML vocabulary for chemical reactions. *J. Chem. Inf. Model*, 46(1):145–157, 2006.
- [19] N. Adams, J. Winter, P. Murray-Rust, and H. Rzepa. Chemical Markup, XML and the World-Wide Web. 8. Polymer Markup Language. *J. Chem. Inf. Model*, 48(11):2118–2128, 2008.
- [20] P. Pilot. version 6.1. *Accelrys: San Diego, CA*, 2008.
- [21] M. Randić and S. Basak. Optimal molecular descriptors based on weighted path numbers. *J. Chem. Inf. Comput. Sci*, 39(2):261–266, 1999.
- [22] A. Llinàs, R. Glen, and J. Goodman. Solubility challenge: can you predict solubilities of 32 molecules using a database of 100 reliable measurements? *Journal of chemical information and modeling*, 48(7):1289–1303, 2008. ISSN 1549-9596.

- [23] W. Bremser. Hose—a novel substructure code. *Analytica Chimica Acta*, 103(4):355–365, 1978. ISSN 0003-2670.
- [24] F. K. Brown. Chapter 35. chemoinformatics: What is it and how does it impact drug discovery. volume 33 of *Annual Reports in Medicinal Chemistry*, pages 375 – 384. Academic Press, 1998.
- [25] J. Bicerano. Prediction of the properties of polymers from their structures. *Polymer Reviews*, 36(1):161–196, 1996. ISSN 1558-3724.
- [26] H. Wiener. Structural determination of paraffin boiling points. *Journal of the American Chemical Society*, 69(1):17–20, 1947. ISSN 0002-7863.
- [27] I. Gutman and N. Trinajstić. Graph theory and molecular orbitals. total φ -electron energy of alternant hydrocarbons. *Chemical Physics Letters*, 17(4):535–538, 1972.
- [28] S. Basak, V. Magnuson, G. Niemi, R. Regal, and G. Veith. Topological indices: their nature, mutual relatedness, and applications. *Mathematical Modelling*, 8:300–305, 1987. ISSN 0270-0255.
- [29] A. Balaban. Chemical graphs. *Theoretical Chemistry Accounts: Theory, Computation, and Modeling (Theoretica Chimica Acta)*, 53(4):355–375, 1979. ISSN 1432-881X.
- [30] L. Kier and L. Hall. The nature of structure-activity relationships and their relation to molecular connectivity. *Eur. J. Med. Chem*, 12(307-312):334, 1977.
- [31] L. H. Hall and L. B. Kier. The molecular connectivity chi indexes and kappa shape indexes in Structure-Property modeling. In *Reviews in Computational Chemistry*, volume 2, pages 367–422. John Wiley & Sons, Inc., Hoboken, NJ, USA, 1991. ISBN 9780470125793.
- [32] H. Morgan. The Generation of a Unique Machine Description for Chemical Structures-A Technique Developed at Chemical Abstracts Service. *Journal of Chemical Documentation*, 5(2):107–113, 1965. ISSN 0021-9576.
- [33] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: An update. *SIGKDD Explorations*, 11(1), 2009.
- [34] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001. ISSN 0885-6125.

- [35] T. Fawcett. An introduction to ROC analysis. *Pattern recognition letters*, 27(8):861–874, 2006. ISSN 0167-8655.
- [36] Y. Zhang, P. Murray-Rust, M. Dove, R. Glen, H. Rzepa, J. Townsend, S. Tyrrell, J. Wakelin, and E. Willighagen. JUMBO—An XML infrastructure for eScience. In *Proceedings of UK e-Science All Hands Meeting*. 2004.
- [37] J. Clark et al. XSL transformations (XSLT) version 1.0. *W3C recommendation*, 16(11), 1999.
- [38] F. Wang, M. Hickner, Y. Kim, T. Zawodzinski, and J. McGrath. Direct polymerization of sulfonated poly (arylene ether sulfone) random (statistical) copolymers: candidates for new proton exchange membranes. *Journal of Membrane Science*, 197(1-2):231–242, 2002. ISSN 0376-7388.
- [39] D. Benoit, C. J. Hawker, E. E. Huang, Z. Lin, and T. P. Russell. One-Step formation of functionalized block copolymers. *Macromolecules*, 33(5):1505–1507, March 2000.
- [40] I. Suárez, M. J. Caballero, and B. Coto. Composition effects on ethylene/propylene copolymers studied by GPC-MALS and GPC-IR. *European Polymer Journal*, 46(1):42–49, January 2010. ISSN 0014-3057.
- [41] Y. Takaeda and K. Yagi. Polymers in Japan. *Polymer News*, 28(11):352–255, 2003.
- [42] Polymers: A property database. URL <http://poly.chemnetbase.com>.
- [43] Polymerlibrary. URL <http://www.polymerlibrary.com/>.
- [44] Polymerweb. URL <http://www.polymerweb.com/>.
- [45] Polymerprocessing. URL <http://www.polymerprocessing.com>.
- [46] R. G. Beaman. Relation between (apparent) second-order transition temperature and melting point. *Journal of Polymer Science*, 9(5):470–472, 1952. ISSN 00223832.
- [47] Polyinfo website. URL http://polymer.nims.go.jp/index_en.html.
- [48] A. Barton. Solubility parameters. *Chemical Reviews*, 75(6):731–753, 1975.

- [49] C. Hansen. *Hansen solubility parameters*. CRC Press Boca Raton, 1999. ISBN 0849315255.
- [50] G. Fytas. Physical Optics of Dynamic Phenomena and Processes in Macromolecular Systems. *27th IUPAC Prague Microsymposium, Walter de Gruyter and Co., West Berlin*, 1984.
- [51] E. J. Merz, L. E. Nielsen, and R. Buchdahl. Influence of molecular weight on the properties of polystyrene. *Industrial & Engineering Chemistry*, 43(6):1396–1401, June 1951.
- [52] W. A. Lee and G. J. Knight. Ratio of the glass transition temperature to the melting point in polymers. *British Polymer Journal*, 2(1):73–80, 1970. ISSN 00071641.
- [53] L. Nelson. *Mechanical Properties of Polymers*. Reinhold, New York, 1962.
- [54] R. Boyer. The relation of transition temperatures to chemical structure in high polymers. *Rubber Chem. Technol.*, 36:1303, 1963.
- [55] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2010. ISBN 3-900051-07-0, URL <http://www.R-project.org>.
- [56] J. M. Chambers. *Statistical Models in S*, chapter 4. Wadsworth & Brooks/Cole, 1992.
- [57] A. Gottwald, D. Pospiech, D. Jehnichen, L. H "außler, P. Friedel, J. Pionteck, M. Stamm, and G. Floudas. Self-Assembly and Viscoelastic Properties of Semifluorinated Polyesters. *Macromolecular chemistry and physics*, 203(5-6):854–861, 2002. ISSN 1521-3935.
- [58] Tagsoup. URL <http://home.ccil.org/~cowan/XML/tagsoup>.
- [59] Xhtml 1.0 the extensible hypertext markup language (second edition). URL <http://www.w3.org/TR/2002/REC-xhtml1-20020801/>.
- [60] eXist. URL <http://exist.sourceforge.net/>.
- [61] Chemical Computing Group, Montreal, Quebec, Canada. *MOE*, 2009.
- [62] D. Knuth. Backus normal form vs. backus naur form. *Communications of the ACM*, 7(12):735–736, 1964. ISSN 0001-0782.

- [63] T. Parr and R. Quong. ANTLR: A predicated-LL (k) parser generator. *Software: Practice and Experience*, 25(7):789–810, 1995. ISSN 1097-024X.
- [64] A. Eisenberg and J. Melton. SQL: 1999, formerly known as SQL3. *ACM SIGMOD Record*, 28(1):138, 1999. ISSN 0163-5808.
- [65] M. R. Berthold, N. Cebron, F. Dill, T. R. Gabriel, T. Kötter, T. Meinl, P. Ohl, C. Sieb, K. Thiel, and B. Wiswedel. KNIME: The Konstanz Information Miner. In *Studies in Classification, Data Analysis, and Knowledge Organization (GfKL 2007)*. Springer, 2007. ISBN 978-3-540-78239-1. ISSN 1431-8814.
- [66] D. M. Lowe, P. T. Corbett, P. Murray-Rust, and R. C. Glen. Chemical name to structure: Opsin, an open source solution. *J. Chem. Inf. Model.*, Forthcoming publication.
- [67] N. O’Boyle, C. Morley, and G. Hutchison. Pybel: a Python wrapper for the OpenBabel cheminformatics toolkit. *Chemistry Central Journal*, 2(1):5, 2008. ISSN 1752-153X.
- [68] Openbabel - opensource chemistry toolkit. URL <http://openbabel.sourceforge.net/>.
- [69] Smarts - a language for describing molecular patterns. URL <http://www.daylight.com/dayhtml/doc/theory/theory.smarts.html>.
- [70] Y. Tomikawa, M. Kimura, and C. Maeda. "Nikkaji Web" has been released. *Journal of Information Processing and Management*, 48(4):220, 2005. ISSN 0021-7298.
- [71] J. Brecher. Name=Struct: a practical approach to the sorry state of Real-Life chemical nomenclature. *J. Chem. Inf. Comput. Sci.*, 39(6):943–950, November 1999. ISSN 1549-9596.
- [72] E. A. Hill. ON a SYSTEM OF INDEXING CHEMICAL LITERATURE; ADOPTED BY THE CLASSIFICATION DIVISION OF THE u. s. PATENT OFFICE. *J. Am. Chem. Soc.*, 22(8):478–494, August 1900.
- [73] S. Stein, S. Heller, and D. Tchekhovski. An open standard for chemical structure representation: the IUPAC Chemical Identifier. In *Proceedings of the 2003 International Chemical Information Conference (Nimes)*, pages 131–143. 2003.

- [74] H. Schoenbacher and A. Stolarz-Isycka. Compilation of radiation damage test data, part 11: Thermosetting and thermoplastic resins. *CERN*, pages 79–08, August 1979.
- [75] N. Adams. Polymer informatics. *Advances in Polymer Science*, 225(1):107–149, 2010.
- [76] J. Xu, B. Chen, Q. Zhang, and B. Guo. Prediction of refractive indices of linear polymers by a four-descriptor QSPR model. *Polymer*, 45(26):8651–8659, December 2004. ISSN 0032-3861.
- [77] B. E. Mattioni and P. C. Jurs. Prediction of glass transition temperatures from monomer and repeat unit structure using computational neural networks. *Journal of Chemical Information and Computer Sciences*, 42(2):232–240, March 2002.
- [78] C. Cao and Y. Lin. Correlation between the glass transition temperatures and repeating unit structure for high molecular weight polymers. *Journal of Chemical Information and Computer Sciences*, 43(2):643–650, March 2003.
- [79] A. R. Katritzky, S. Sild, and M. Karelson. Correlation and prediction of the refractive indices of polymers by QSPR. *Journal of Chemical Information and Computer Sciences*, 38(6):1171–1176, November 1998.
- [80] A. Afantitis, G. Melagraki, H. Sarimveis, P. A. Koutentis, J. Markopoulos, and O. Igglessi-Markopoulou. Prediction of intrinsic viscosity in polymer-solvent combinations using a QSPR model. *Polymer*, 47(9):3240 – 3248, 2006. ISSN 0032-3861.
- [81] R. A. Campo-Arnáiz, M. A. Rodríguez-Pérez, B. Calvo, and J. A. de Saja. Extinction coefficient of polyolefin foams. *Journal of Polymer Science Part B: Polymer Physics*, 43(13):1608–1617, 2005. ISSN 0887-6266.
- [82] G. K. Zipf. *Selected Studies of the Principle of Relative Frequency in Language*. Harvard university press, 1932.
- [83] M. Newman. Power laws, Pareto distributions and Zipf’s law. *Contemporary physics*, 46(5):323–351, 2005. ISSN 0010-7514.
- [84] H. Drucker, C. Burges, L. Kaufman, A. Smola, and V. Vapnik. Support vector regression machines. *Advances in neural information processing systems*, pages 155–161, 1997. ISSN 1049-5258.

- [85] C. Chang and C. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
- [86] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995. ISSN 0885-6125.
- [87] A. Smola and B. Schölkopf. A tutorial on support vector regression. *Statistics and computing*, 14(3):199–222, 2004. ISSN 0960-3174.
- [88] S. Shevade, S. Keerthi, C. Bhattacharyya, and K. Murthy. Improvements to the smo algorithm for svm regression. *Neural Networks, IEEE Transactions on*, 11(5):1188–1193, September 2000. ISSN 1045-9227.
- [89] A. Afantitis, G. Melagraki, K. Makridima, A. Alexandridis, H. Sarimveis, and O. Iglessi-Markopoulou. Prediction of high weight polymers glass transition temperature using RBF neural networks. *Journal of molecular structure: THEOCHEM*, 716(1-3):193–198, 2005. ISSN 0166-1280.
- [90] A. R. Katritzky, S. Sild, V. Lobanov, and M. Karelson. Quantitative Structure-Property relationship (QSPR) correlation of glass transition temperatures of high molecular weight polymers. *Journal of Chemical Information and Computer Sciences*, 38(2):300–304, March 1998.
- [91] J. Bicerano. *Prediction of Polymer Properties*. Marcel Dekker, 2002. ISBN 0-8247-0821-0.
- [92] D. Lide and H. Frederikse. *CRC Handbook of Chemistry and Physics: A Ready-reference Book of Chemical and Physical Data:[1913-1995]*. CRC Press, 1994. ISBN 084930475X.
- [93] P. Labute. Moe molar refractivity model, 1998. Source code in MOE/lib/avl/quasar.svl/q_mref.svl.
- [94] S. Wildman and G. Crippen. Prediction of physicochemical parameters by atomic contributions. *J. Chem. Inf. Comput. Sci*, 39(5):868–873, 1999.
- [95] P. Labute. Moe logp(octanol/water) model, 1998. Source code in MOE/lib/avl/quasar.svl/q_logp.svl.
- [96] T. Hou, K. Xia, W. Zhang, and X. Xu. ADME evaluation in drug discovery. 4. Prediction of aqueous solubility based on atom contribution approach. *J. Chem. Inf. Comput. Sci*, 44(1):266–275, 2004.

- [97] T. I. Oprea. Property distribution of drug-related chemical databases. *Journal of Computer-Aided Molecular Design*, 14:251–264, 2000. ISSN 0920-654X.
- [98] P. Ertl, B. Rohde, and P. Selzer. Fast calculation of molecular polar surface area as a sum of fragment-based contributions and its application to the prediction of drug transport properties. *J. Med. Chem*, 43(20):3714–3717, 2000.
- [99] C. Lipinski, F. Lombardo, B. Dominy, and P. Feeney. Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings. *Advanced Drug Delivery Reviews*, 23(1-3):3–25, 1997. ISSN 0169-409X.
- [100] A. Balaban. Highly discriminating distance-based topological index. *Chemical Physics Letters*, 89(5):399–404, 1982. ISSN 0009-2614.
- [101] R. Pearlman and K. Smith. Novel software tools for chemical diversity. *Perspectives in Drug Discovery and Design*, 9:339–353, 1998. ISSN 0928-2866.
- [102] M. Petitjean. Applications of the radius-diameter diagram to the classification of topological and geometrical shapes of chemical compounds. *Journal of Chemical Information and Computer Sciences*, 32(4):331–337, 1992. ISSN 0095-2338.